RICE UNIVERSITY

On the Modeling of Signaling Networks with Petri Nets

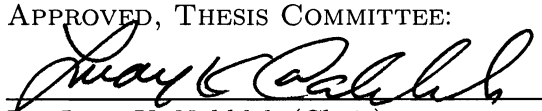
by

Natalie Berestovsky

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

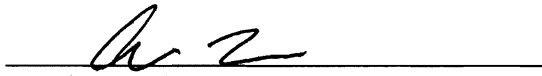
Master of Science

APPROVED, THESIS COMMITTEE:

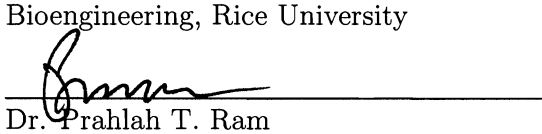


Dr. Luay K. Nakhleh (Chair)
Associate Professor,
Computer Science, Rice University

Dr. Lydia E. Kavraki
Professor,
Computer Science, Rice University



Dr. Amina A. Qutub
Assistant Professor,
Bioengineering, Rice University



Dr. Prahlah T. Ram
Associate Professor, Systems Biology,
The UT MD Anderson Cancer Center

HOUSTON, TEXAS
APRIL 2011

Abstract

On the Modeling of Signaling Networks with Petri Nets

by Natalie Berestovsky

The whole-cell behavior arises from the interplay among signaling, metabolic, and regulatory processes. Proper modeling of the overall function requires accurate interpretations of each component. The highly concurrent nature of the inner-cell interactions motivates the use of Petri nets as a framework for the whole-cell modeling. Petri nets have been successfully used in modeling of metabolic pathways, as it allows for a straightforward mapping from its stoichiometric matrix to the Petri net structure. The Boolean interpretation and modeling of transcription regulation networks also lends itself easily to Petri net modeling. However, Petri net modeling of signal transduction networks has been largely lacking, with the exception of simple ad hoc applications to specific signaling pathways. In this thesis, I investigate the applicability of Petri nets to modeling of signaling networks, by systematically analyzing initial token assignments, firing strategies, and robustness to errors and abstractions in the estimates of molecule concentrations and reaction rates.

Acknowledgements

When we graduate college, some of us go to work, happily saying good bye to the sleepless nights before the deadlines and the weekends filled with homework, others go to grad school because they just cannot get enough of it. For me, applying to graduate school seemed like the natural next thing to do. However, while I was visiting different universities I started doubting whether it was what I really wanted...Until, while visiting Rice, over lunch I started chatting with Luay Nakhleh about the research in the field of bioinformatics. I cannot explain it, but it was one those "eureka" moments, when you know this is *it*.

First and foremost, I would like to thank my advisor, Prof. Luay Nakhleh. His enthusiasm and excitement about the problems and the research opportunities in bioinformatics inspired me to pursue this field. His guidance, understanding, support and expertise helped me to get to this milestone in my academic career. I feel truly privileged to have the opportunity to work with Luay during my grad years. I cannot count the number of times I was told by other students that I am lucky to have him as my advisor. And, I am, truly, lucky.

I would like to thank my committee Prof. Prahlad Ram from UT MD Anderson, Prof. Lydia Kavraki from Rice Computer science and Dr. Amina Qutub from Rice Bioengineering. Thank you for taking time to review my work and provide useful feedback.

I would like to express my gratitude to the sources of funding for this work: Seed Funding Program Collaborative Advances in Biomedical Computing (CABC), funded by the John and Ann Doerr Fund for Computational Biomedicine, and the National

Cancer Institute (Grant Number R01CA125109).

I would like to thank my family for being there for me at the hardest moments, and believing in me. Thank you to my parents Svetlana and Sergey Yudin for instilling in me the love for math and science from the very childhood. Thank you to my brother Andrey Yudin for being my best friend. Being very close in age, we have gone hand-in-hand throughout our academic paths, and the voyage is always easier if you are not alone. Thank you to my husband Dmitiy Berestovsky for his love, support, understanding and solicitude. Thank you for making me soups, keeping me sane, and taking care of so many things!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the world of targeted therapeutics, in silico simulations are becoming increasingly important. They unveil the anticipated behavior of the system under desired experimental conditions to assist better "wet" experimental design. The complexity of the intracellular processes makes the construction of dynamical models a non-trivial task. The characterization of system's properties arises from the whole-cell function which is comprised of *signaling*, *metabolic* and *regulation* processes (see Figure 1.1). In *modeling* of these systems one seeks a chemically accurate representation of all components of biochemical events and incorporation of the functional relationships between them. Each component has been successfully modeled in isolation via various dynamic and structural analyses. However, in the search of the integrated model it is important to identify a *single framework* that embodies each component equivalently well and provides ways to capture their interdependencies within one framework.

When seeking an executable model to represent a biochemical process, one needs

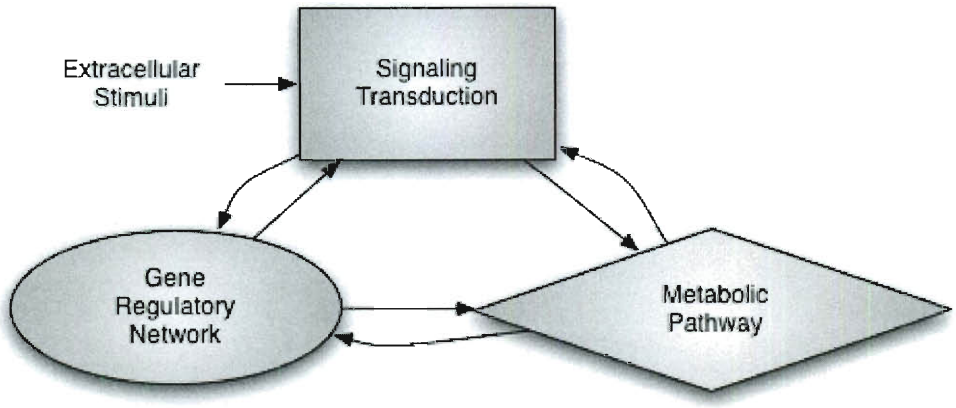


Figure 1.1: Interconnectivity between signaling, metabolic, and regulatory networks. Each of the three main functional modules of the intracellular behavior have been modeled in isolation. However, on the way to the integrated modeling it is important to consider the interplay between those components.

to consider two factors: the *modeling framework*, and the *execution strategy*. In [1], Fisher *et al.* distinguishes between two types of modeling frameworks - *mathematical* and *computational*. The former representation is based on the mathematical relationships between quantities that are derived from biochemical kinetics. The latter one is a representation of the state machine, which relates different qualitative configurations ("state") to each other. Both frameworks have been used to represent components of the intracellular process. The most common way of mathematical modeling is representing a process of interest in terms of ordinary differential equations (ODE). In this approach, the concentrations of species (i.e., metabolites, signaling proteins) are described by mass-balance equations that incorporate kinetic details of reaction mechanisms and their associated kinetic parameters. Solving the set of ODEs over

time generates the dynamics of the system. On the side of computational approaches, *Petri nets*[1] stand out as one of the most frequently used modeling frameworks. In this graphical model, there exist two types of nodes - places and transitions. Places store the state of the species, and transitions encode the relationships between them. Petri nets, when executed, move through the states of the system, recording the overall condition at each step. More detailed description of both modeling frameworks is provided in Chapter 2.

When it comes to executing the model, we distinguish between two execution strategies - *deterministic* and *stochastic*. With regards to the chemical kinetics, both numerical methods have strong physical basis. When executed, first one handles the time evolution as a continuous, wholly predictable process, whereas the second one treats the system as a Markovian random-walk process in the N-dimensional space of the molecular population of the N species [2]. Both types of dynamic models and their execution formalisms are discussed in chapter 3.

On the way to modeling the whole-cell function, the foremost task is to find a single framework that defines each component equally well, as well as has the flexibility to encode their interconnections. In this work, I show that the computational framework of Petri nets is suitable for modeling the full-cell behavior. First, I review the successful use of Petri net structure in modeling of metabolic and regulatory networks. Then, I investigate the use of Pertri nets in signaling pathways. Particularly, I focus on the robustness of Petri nets to the parameter inaccuracies frequently

[1]Not to be confused with *Petri dish* - a shallow glass or plastic cylindrical lidded dish that biologists use to culture cells or small moss plants, invented by Julius Richard Petri.

associated with signaling. The robustness testing procedure is described in Chapter 4.

1.1 Motivation

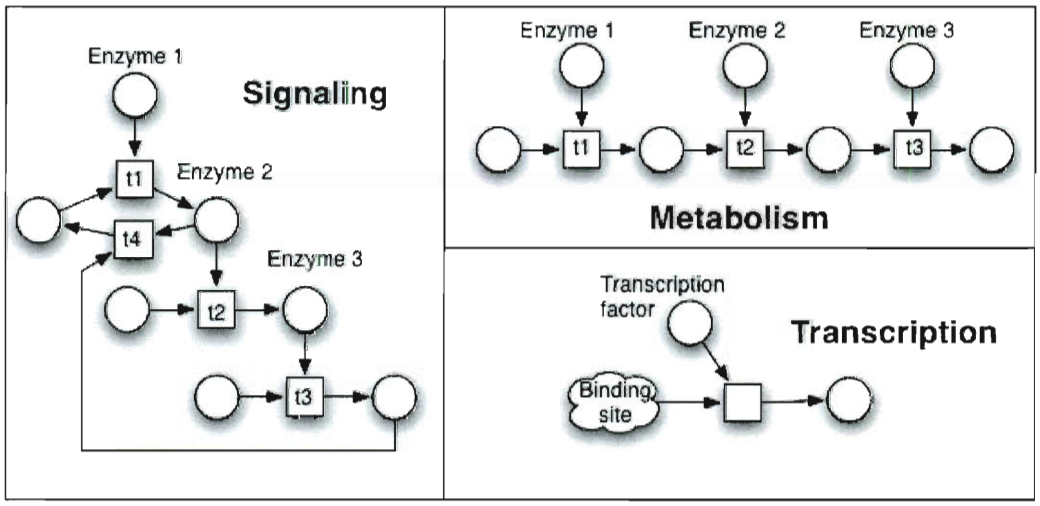


Figure 1.2: Petri net representation of three main components of the cell function. *Circles* represent species in the system. *Squares* represent reactions between these species. *Arrows* towards the squares come from the substrates of the reactions, and arrow out of the squares point to the products. Metabolism is a serial process in which the product of one reaction becomes a substrate of the other. Signaling is highly interleaved process with possible feed-forward and -backward loops. Regulation is a switch-like processes that activates a gene when both the binding site and the transcription factor nodes are in active state. While modeling and analyzing metabolic and regulatory networks using Petri nets have been commonly done in the research community, work remains to be done on modeling and analyzing signaling networks by Petri nets, which is the focus of this thesis.

The *big picture motivation* for this work is to show that Petri net computational model provides a good framework for modeling whole-cell behavior. The advantage

of Petri nets over other modeling methods is that (1) they employ a very simple model that has an intuative graphical representation, (2) enable both deterministic and stochastic analysis, (3) and can combine system representations at coarse- and fine-grained levels incorporating components with different time scales. Simple Petri net representations of the three main components of whole-cell function are shown in Figure 1.2. I now explain why Petri net is a good modeling framework for each component.

Metabolism is the processes through which living systems acquire and utilize the free energy they need. In this process many catalyst chemical reactions take place in a serial manner, where the product of one reaction is usually a substrate of another reaction. In metabolic pathways the chemical reactions rates are given by their stoichiometric equations. Usually, the rates, the metabolite concentrations, and

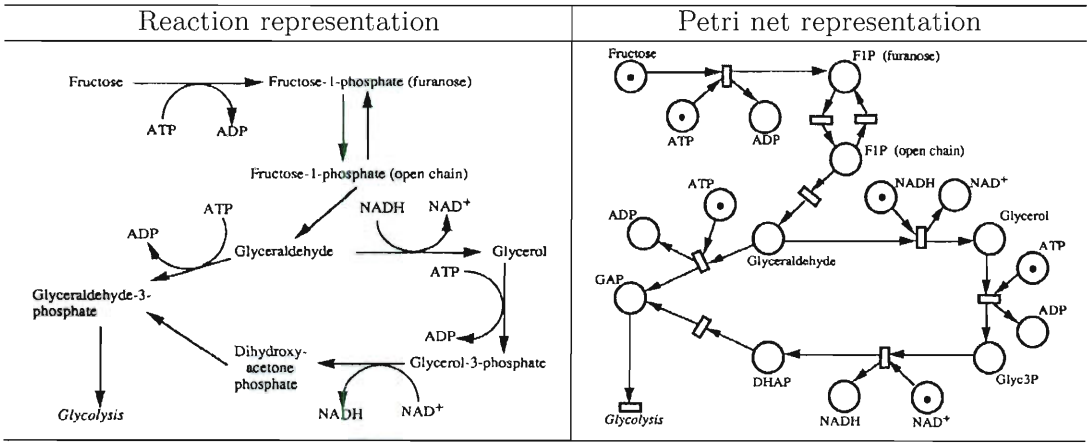
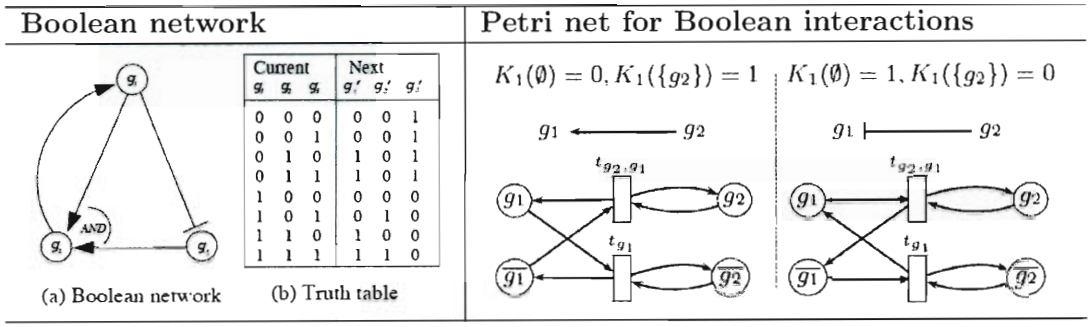


Figure 1.3: Example of a Petri net representation of metabolic network. All the reactions are catalyzed by enzyme. Left panel shows the reaction associated with the processes. Right panel shows its PN representation. There is almost one-to-one mapping between places and metabolites. Adapted from [3].

the enzyme concentrations are known [4]. Additionally, despite of the complexity of the internal reactions, under normal conditions living systems maintain a steady state, resulting in the conservation in the system. The commonly used mathematical model of flux balance analysis (FBA) is based on the steady state analysis and requires only stoichiometric coefficients of the reactions. This constraints-based framework has been used successfully to predict steady state concentrations of the system [5]. The sequentiality of the process along with the known constants create fitting modeling environment for Petri nets. One of the first successful Petri net-based models of metabolism was devised by Reddy *et al.* in [3, 6]. Figure 1.3 depicts Reddy's model from [3]. In the left panel, I show a graphical representation of the reactions associated with the metabolic process. In the right panel I show its Petri net model. The sequential nature of the process along with its innate stoichometry creates favorable conditions for modeling with the most basic, qulivative Petri nets (Section 2.2.1) [7]. Over the recent years, the various types of Petri nets have been extensively used in modeling different metabolic applications [4, 8, 9, 10]. Further, in the metabolism-related case study, Popova-Zeugmann *et al.* introduces a notion of *time* to the Petri net model (Section 2.2.2), making it executable and allowing to examine the transient behavior of the system [11].

Regulation of gene the expression includes the processes by which cells turn the information in genes into the gene products. Since 1969, when Kauffman proposed the use of Boolean network for regulation modeling, the method has been widely used [12]. Over the years, with the steady increase in the amount of data on genetic regulation, Boolean networks became a common strategy for regulation modeling



Current			Next		
g_1	g_2	g_3	g_1'	g_2'	g_3'
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	0

Figure 1.4: Example of a Boolean network and Petri net representations of Boolean interactions: The left panel shows an example of a Boolean network with its transition truth table. The right panel shows an example of genetic activation and genetic inhibition converted into Petri nets (these interactions are unrelated to the nodes on the left panel). Finction $K_i(Z)$ represents the Boolean logic associated with node i and the input set Z. Left panel is adopted from [16]. Right panel is adopted from [17].

[13, 14, 15]. In the left panel in Figure 1.4, I show an example of a simple Boolean network model. Each node represent a gene, whose state can be either 0 or 1 (*off* or *on*). The transitions comprise the Boolean logic the determine the state of each node. The transition execute all at once, resulting in the state change according to the table shown in (b). In [17, 18] Chaouiya initiated the idea of using Petri nets to model genetic regulation systems. In the right panel of Figure 1.4, I show an example of gene activation and gene inhibition represented using Petri nets (these interactions are unrelated to the Boolean network on the left). For each gene g_i, we define a complementary place $\bar{g}_i$, such that the sum of tokens in places g_i and $\bar{g}_i$ equals to 1. If the token is in g_i or $\bar{g}_i$, we consider it is *on* and *off* respectively. For each gene g_i, there exists a logical function $K_i(Z)$, where the input Z is a set of operating incoming interactions. The resulted model addresses a number of shortcomings associated with the Boolean networks, such as allowing the variables to have more than the two

values (more than two nodes can represent the gene and its activity), and transitions between the states to occur asynchronously.

Further, Steggles and Banks expanded on this idea. They detail a process for automatically constructing these models using logic minimization to translate the Boolean terms into appropriate Petri net control structures [16, 19]. In [20] Jong provides an extensive review on the methods for regulation modeling. In recent years, further development on the use of Petri nets in regulatory networks has been done using successful *hybrid* approaches [21, 22] which would allow timed executions.

Signaling pathways are highly interleaved and parametrized. The concentrations on nodes are much more refined and should be represented by the range of values. The interactions in signaling pathways happen in parallel with different rates and with possible feedforward and -backward loop without clear stoichiometry mapping. The network topology of interactions is frequently incomplete and rate parameters are generally hard to estimate accurately. Despite these limitations, Petri nets have shown to be applicable in the signaling pathways as well. In [23] and [24], timed classic qualitative Petri nets have shown to be exploited (Section 2.2.1) to model signaling pathways. In both works the issue of parameters is the central concern on the way of acquiring accurate results.

Petri net is a good framework for modeling a whole cell behavior. For metabolic networks there is almost a one-to-one correspondence between stoichiometry of the metabolic reactions and the Petri net structure. For regulatory networks, there exist conversion techniques from widely accepted Boolean networks to the Petri net struc-

tures. For signaling networks, while Petri net looks as a promising approach, more investigation is need on its applicability [25]. In this work I investigate the use of Petri net as a modeling framework for signaling networks. Particularly I address the issues that arise due to the frequent inaccuracies and incompleteness associated with signaling models.

1.2 My work

In Figure 1.5, I show the flowchart highlighting the main components of my work. The available *data* is the main inspiration for this work. Biologist hand-devise the graphical descriptions of signaling pathways based on the knowledge of the studied interactions. Further, they can experimentally derive the parameters associated with the system. This final description becomes the model of the signaling pathway. In order to make predictions based no this model, one can put it in the context of the executable *framework*. In my work, I use Petri nets as the computational tool of choice. The *contribution* of my work is in investigating the applicability of the Petri net framework to the modeling of signaling pathways. I examine how the elements of biological description of signaling network can be assigned to the Petri net components. First, I describe how to accurately store the signaling network topology in the Petri net structure. Then I explain how the parameterization information can be extracted from the experimental data and applied to the Petri net. Further, to address the dynamics of signaling pathways, I investigate the two simulation strategies for executing Petri net: continuous (deterministic) and stochastic. After constructing

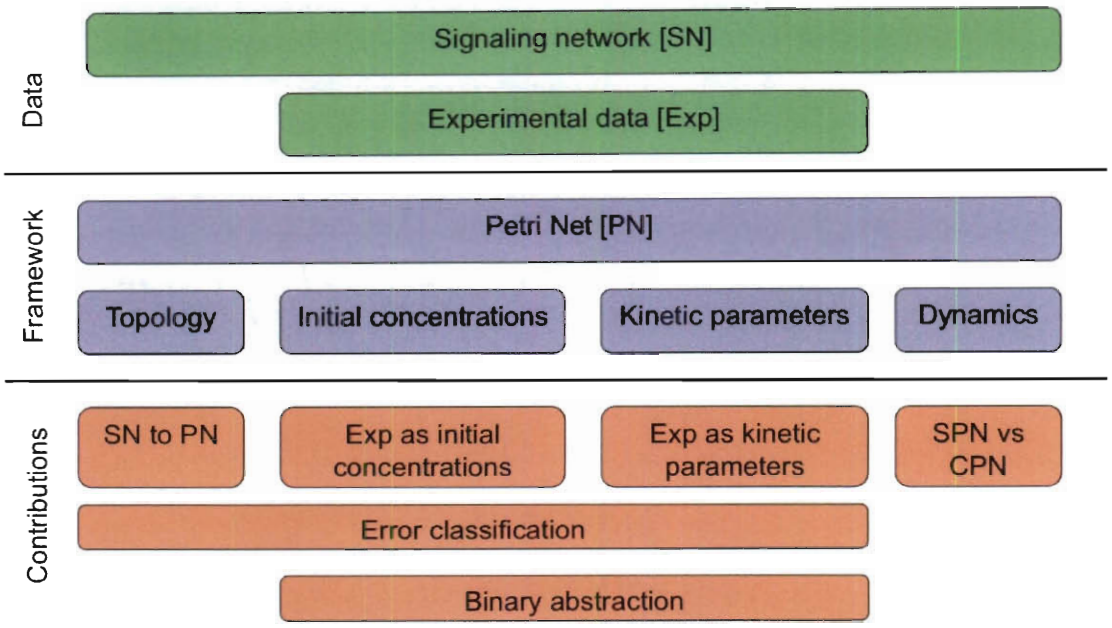


Figure 1.5: Thesis flowchart. *Data:* Signaling network topologies and the abundance of the experimental data associated with them is the main inspiration of this work. *Framework:* Petri net is computation tool of choice. *Contributions:* Investigating the applicability of the Petri net framework to the models of signaling pathways. First I examine how each element of biological description of signaling pathway maps onto the PN components. Second, I test the "signaling Petri net" to the potential inaccuracies that could have been present in the original biological description. Lastly, I create an abstracted model (in terms of parameters and initial concentrations) and investigate how much one can see just from the topology of the network.

a "signaling Petri net", I test if for the robustness to the potential inaccuracies that could have been present in the original biological description. The error robustness testing covers potential inaccuracies in kinetic parameter and initial concentrations, along with potential incompleteness of the topology. Lastly, I investigate how much of the dynamic information can be extracted just from the topology of the network. To address this, I create a binary abstracted model, I reassign the full range of the kinetic

value with either one 'low' and one 'high' parameters (creating binary assignment). Then I replace all the initially present concentrations with the same number of tokens, and absent concentrations are set to zero. The binary abstracted model is then compared to the original one in terms of the dynamic outcomes. For determining the error between two systems in both robustness testing and binary abstraction, I choose two measures: (a) time series tail comparison and (b) time series total comparison. Based on these error values and the observed behaviors, I create the classification scheme to predict the potential changes that could have occurred due to the error. Additionally, on the way to obtaining the results, I do the extensive literature search on the tool and methods I put under investigation.

In Chapter 2, I examine two modeling approaches: *mathematical* and *computational.* I describe how a set of ODEs is derived from the laws of biochemical kinetics for mathematical modeling. I introduce Petri nets as an example of the computation framework.

In Chapter 3, I explain the Petri nets' dynamic capabilities. Since ODE modeling is a trusted method for attaining deterministic time series, in section 3.1 I show how dynamic ODE system, can be converted into the Petri net model preserving all the kinetic values. I, then, show stochastic formulation alternative and show the accuracy of both methods with respect to each other and to the experimental data.

Further, in Chapter 4, I describe the systematic testing procedure for investigating robustness of Petri nets to the potential inaccuracies present in the signaling pathways. In this chapter, I also investigate the different distance measures for comparing the outcomes of time series simulations. I identify the most appropriate distance for

both tail (potential steady-state) and total (tune series data) measurements.

In Chapter 5, I present the results of the proposed work. I examine how each source of error affects the simulation outcomes. I examine two signaling network systems with different dimensions. The smaller one, β2AR, is highly interconnected network consisting of 10 species and 23 reactions. The other one, ERK/MEK, is larger but less dense, containing 40 species and 47 reactions. I acquire tail and total errors by comparing systems with inaccuracies to the original execution. Using only these values I predict how the error affected individual species in the simulation. I classify the effects into three categories: (i) the species shifted in its steady state, but followed its overall trend, (ii) the species completely lost its transient trend and potentially did not reach the steady state in the allocated amount of the experiment time (least favorable outcome) and (iii) the species temporarily shifted in transient behavior by eventually achieve accurate steady state. Further, I validate my predictions via visually analyzing the changes that have occurred due to the error. The inspections show that my classifications are highly accurate. Additionally, my findings show that different amounts of error in the network mostly create shifting in trends, and only for small number on species it causes complete deviation from the original behavior.

In Chapter 6, I summarize the results, review the contributions that come out of this thesis, and provide directions for the future work.

Chapter 2

Modeling

A biological model begins as graphical delineation of the known interactions within the system. It is then augmented based on the findings form the experimental observation [26]. This description becomes a tremendous help in creating an executable model to represent the system. In this chapter I discuss two type of models: *mathematical* in section 2.1 and *computational* in section 2.2.

2.1 Mathematical modeling

A mathematical model is a formal model whose primary semantics is denotational; that is, the model describes by equations (ODEs) a relationship between quantities and how they change over time [1]. Solving these equations over time approximates the dynamics of the system. In order to achieve accurate dynamics, the ODE systems requires a complete set of kinetic parameters. In this lies the main limitation of this

method, since while kinetic data is available for some reactions, for the vast majority of the pathways kinetic parameters have not been identified [27]. In this section I quickly review the two classical principles of chemical reaction kinetics: *The Law of Mass Action* and *Michaelis-Menten Kinetics.* We show how these concepts are used to derive ODE-based models to predict dynamic behavior of the system.

Mass action kinetics states that the reaction rate of conversation of masses in a chemical reaction is proportional to the product of the masses of the reacting substances. Considering a generic reaction $mA + nB \xrightarrow{k} C$ with reactants A and B and stoichiometric coefficients m and n, the reaction rate is given by

$$r = k[A]^m[B]^n, \tag{2.1}$$

where r denotes the reaction rate, $[A]$ and $[B]$ the concentrations of substrates A and B respectively, and k the *kinetic* or *rate constant.*

Given this definition, we can derive a set of ODEs describing the concentration of participating species for *first-order* reaction $A \xrightarrow{k} B$:

$$\begin{aligned} \frac{d[A]}{dt} &= -r = -k[A], \\ \frac{d[B]}{dt} &= +r = +k[A]. \end{aligned} \tag{2.2}$$

Similarly, for the *second-order* or/and *bimolecular* reaction $2A + B \xrightarrow{k} C$:

$$\begin{aligned}
\frac{d[A]}{dt} &= -r = -k[A]^2[B], \\
\frac{d[B]}{dt} &= -r = -k[A]^2[B], \\
\frac{d[C]}{dt} &= +r = +k[A]^2[B].
\end{aligned} \tag{2.3}$$

We can also derive ODEs for the *reversible* reaction $A + B \underset{k_b}{\overset{k_f}{\rightleftharpoons}} C$, where we have to consider two kinetic constants, one for the forward reaction, one for the backward reaction:

$$\begin{aligned}
\frac{d[A]}{dt} &= -r_f + r_b = -k_f[A][B] + k_b[C], \\
\frac{d[B]}{dt} &= -r_f + r_b = -k_f[A][B] + k_b[C], \\
\frac{d[C]}{dt} &= +r_f - r_b = +k_f[A][B] - k_b[C].
\end{aligned} \tag{2.4}$$

The time courses of dynamic behavior can be obtained by integrating these ODEs.

Michaelis-Menten kinetics considers the simple enzymatic reaction $S \xrightarrow{E} P$ where substrate S is catalyzed by enzyme E to make product P. The catalytic rate v is given by:

$$v = \frac{V_{max}[S]}{[S] + K_m}, \tag{2.5}$$

where V_{max} is the maximum reaction velocity and K_m, the *Michaelis constant* - the concentration of the substrate at which the reaction rate is half its maximum value. With the total enzyme concentration $[E_T]$ given by $k_{cat} = V_{max}/[E_T]$ we are able to

describe the ODEs for this system:

$$\frac{d}{dt}[S] = -\frac{d}{dt}[P] = -k_{cat} \cdot [E_T] \cdot \frac{[S]}{K_m + [S]}. \tag{2.6}$$

The main limitation of Michaelis-Menten is that makes a few assumptions: (1) the concentration of product is (close to) zero, (2) No product reverts to the initial substrate and (3) $[E_T] \ll [S]$. Even though these are reasonable assumption for ezyme assays in the test tube, in most metabolic pathways *in vivo* assumptions 1 and 2 don't hold and none are correct for signaling pathways.

However, using the mass action law, we can take into the account the mechanism by which the enzyme acts, producing a one substrate reaction with no backwards reaction effects:

$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{k_2} P + E.$$

We can now relate this model to the original Mechaelis-Menten definition by setting:

$$K_m = \frac{k_{-1} + k_2}{k_1} \text{ where } k_2 = k_{cat} = \frac{V_{max}}{[E_T]} \tag{2.7}$$

We can now derive a set of ODEs for this system:

$$\begin{aligned}
\frac{d[S]}{dt} &= -k_1[E][S] + k_{-1}[ES], \\
\frac{d[E]}{dt} &= -k_1[E][S] + k_{-1}[ES] + k_2[ES], \\
\frac{d[ES]}{dt} &= +k_1[E][S] - k_{-1}[ES] - k_2[ES], \\
\frac{d[P]}{dt} &= +k_2[ES] = v.
\end{aligned} \tag{2.8}$$

These differential equations can be uniquely defined by the Petri Net structure and executed as a continuous Petri net (described in section 3.1.1).

2.2 Computational modeling

A computational model is a formal model whose primary semantics is operational; that is, the model prescribes a sequence of steps or instructions that can be executed by an abstract machine, which can be implemented on a real computer [1]. A choice of underlying data structure can have a tremendous impact of the outcomes of the modeling.

Petri nets (PN) provide a graphical notation for the formal description of the dynamic behavior of the systems. Although, Petri nets have been used for the modeling of computer systems and communication networks since the 1960s [28], they have recently emerged as a promising tool for modeling and analysis of molecular networks. PN methodology seems to be natural choice for modeling biochemical networks, since they share three distinct characteristics. They are (1) inherently bipartite, (2) in-

herently concurrent, and (3) inherently stochastic. The property of being bipartite manifests itself in correspondence between species and their interactions with places and transitions of the Petri nets. The properties of stochasticity and concurrency of biochemical interactions can be modeled with common executable extensions, such as stochastic or continuous Petri nets, both discussed in Chapter 3.

2.2.1 Petri nets basics

A Petri net is a *directed bipartite graph* with two different types of nodes: places and transitions. Generally, places are represented with circle-like shapes and transitions with rectangular shapes. Places can contain *tokens*, represented by a positive rational number and corresponding to concentrations. The state of the system is represented by allocation of tokens over all the places and is called *marking*. It is represented by a vector of numbers the length of which corresponds to the number of places. The initial assignment of tokens is called *initial marking* and represents the starting state of the system.

Transitions and places are connected by directed arcs. *Input places* are the places from which arcs point to the transitions; *output places* are those for which arcs point from the transition. In essence, Petri net's components represent a link between biochemical concepts and theoretical abstractions (see Table 2.1). The simulation of the PN is governed by the *firing* of *enabled* transitions. The transition is enabled when all its input places contain at least the required number of tokens (defined by the inscriptions assigned to the arcs). For the details of Petri net theory, see [29, 30, 31].

The most basic type of Petri net is discrete or qualitative Petri net (QPN). See

Theoretical abstraction	Petri net component	Biochemical process
Atomic action	Transition	Chemical reaction
Local Conditions	Places	Chemical compounds
Multiplicities	Arc weight	Stoichiometric relations
Condition's state	Tokens in the place	Available amount (e.g. mols)
System state	Marking	Compounds distribution

Table 2.1: Petri net link between theoretical abstractions and biochemical processes.

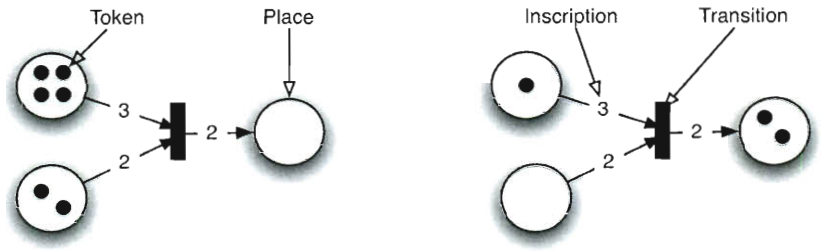


Figure 2.1: Simple firing of a transition. The left image shows the state of PN before the firing of the transition and the right image shows the state immediately after. The transition removes the number of tokens indicated by the inscription of the arcs from the input places and deposits the inscripted number of tokens into the output place. This transition is no longer enabled in the state shown on the right.

Figure 2.1 for a pictorial representation. The sequential firing of enabled transitions is referred to as the *token game* and transforms the system from one state to another. The set of all possible states of the net, given an initial marking, is called the *reachability set* and is peculiar to the initial marking. When analyzing QPN, one can build a reachability set to potentially identify the system's steady state. However, in computational modeling, we are interesting to execute our model as a sequence of steps, building its dynamics as we go. In the following section we discuss how this

coulde accomplished.

2.2.2 Timed Petri Nets

The timed Petri net (TPN) is an intermediate between the purely qualitative approach (QPN) and quantitative models. Given a classical PN, we introduce a time interval $[a_t, b_t]$ associated with each transition t, where a_t and b_t are relative to the time when t last fired. When t becomes enabled, it cannot fire until a_t time units have elapsed, and it must fire no later than b_t time units, unless it becomes disabled by the firing of another transition.

According to the interpretation of Popova-Zeugmann *at al*, a given TPN can now be characterized by a state $z = (m, h)$, where m is the place marking vector as described in the previous section and h is the transitions time marking vector. Analogous to notation $m(p)$ giving the number of tokens in place p, notation $h(t)$ shows the time elapsed since the transition t most recently enabled, or is `null` otherwise (meaning the transition didn't get a chance to fire in interval $[a_t, b_t]$). The time units are represented by the real numbers, whereas interval bounds are rational numbers. As the time units elapse, we fire the transition that has the closest deadline, one a time. Of special interest is the so-called "integer" state. A state $z = (m, t)$ is an "integer" state, iff $h(t)$ is an integer or `null` for each t [11]. This would refer to the steady-state of the simulated metabolic pathway. Use of this model is also demonstrated in [23] with the alternative method of defining time delays and application in signaling transduction. Additionally, stochastic Petri net, described in Chapter 3, is another interpretation of the timed Petri net.

2.2.3 Modeling with Petri nets

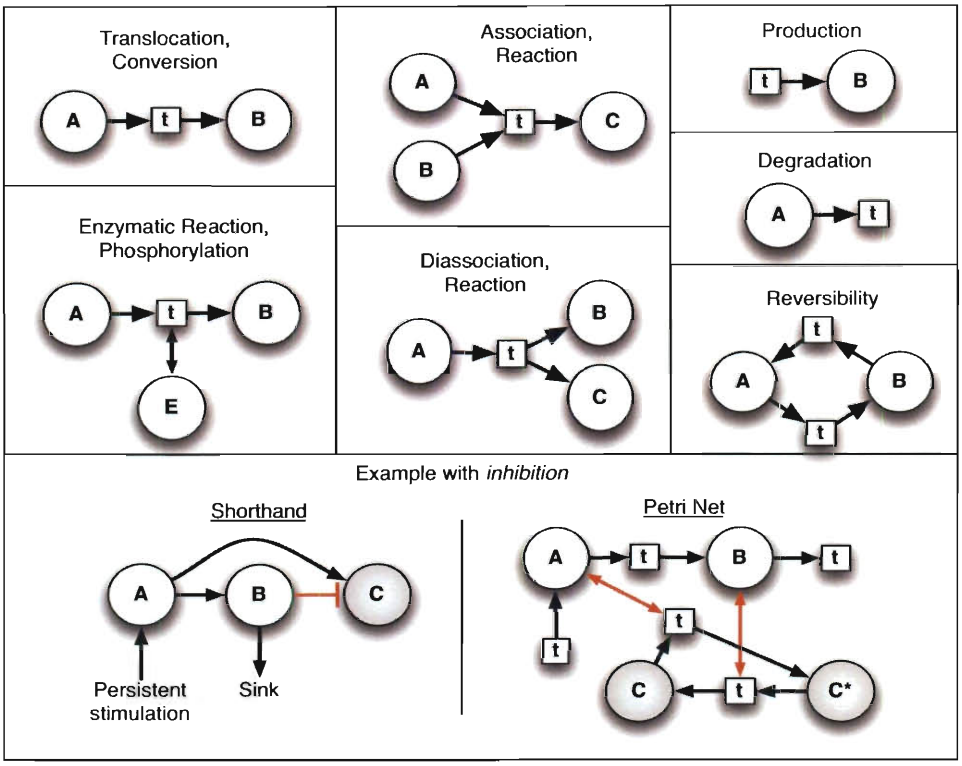


Figure 2.2: Petri net representations of selected biochemical interactions. Single head arrow shows the direction of token flow if the transition is executed. The double-head arrow signifies a read arc, meaning that tokens are removed and put right back in if the reaction is executed (therefore the precense of tokens in this compound is necessary for the reaction to be enabled). Notice, that the transition that represents the production is always enabled, since it has no input.

The systems that are particularly suitable for PN modeling are sometimes refereed to as *Condition/Events* nets. The places identify the condition that make up the state of the system, and the events occur under desired conditions and modify the state of the system [30]. In Figure 2.2 we show the examples of basic PN representations of the

interaction that occur in signaling pathways. While most of the conversions are rather intuitive, one that requires more attention is the process of modeling *inhibition*[1]. In *inhibition* example on Figure 2.2, protein B inhibits C, meaning that it converts phosphorylated form of C into its inactive state. To model this, we break down C into two states: inactive (C) and phosphorylated (C^*). Now, A (an activator) is an enzyme in the phosphorylation reaction, whereas B (an inhibitor) is an enzyme in the de phosphorylation reaction.

In this chapter we explained how mathematical, ODE-based models can be derived from classical chemical kinetics. We, also, showed how Petri net is a good candidate for a computational executable model. In the rest of this thesis we discuss modeling in regards to its dynamics only, referring to ODEs when talking about the mathematical model and Petri nets to represent computational model. In the next chapter, we discuss two model execution strategies - *deterministic* and *stochastic* and how they can be used on above models.

[1] An *inhibitory* arc exists as a common extension of PN formalism. By its definition, the *inhibitory* arc from place p_i to transition t modifies the enabling rule in the sense that the transition can fires only if place p_i does not contain any tokens. This is counterintuitive to the biochemical inhibition, where the highly activated protein (containing tokens) deactivates another protein. Therefore, this arc is not used in this thesis.

Chapter 3

Simulation strategies

In the previous chapter, we discussed two types of structural modeling, *mathematical* and *computational*. In order to produce the dynamic profiles for each system, we can use one of the two execution strategies: *deterministic* or *stochastic*. Both strategies can be expressed in the mathematical model via ODEs and Stochastic ODEs. Section 3.1 provides the steps for achieving deterministic solution in the ODE model. In Section 3.1.1 I provide a description of deterministic simulation in the context of Petri nets via the Continuous Petri nets (CPNs). While the deterministic approach has been widely accepted and used, in 1979 Gillespie proposed the framework of stochastic formulation of chemical kinetics. In [32], Gillespie makes two principle points that make stochastic simulations especially attractive. First, he provides a stronger physical basis for this approach. He claims that, while ODE approach cannot be denied, it evidently assumes that the time evolution of a chemically reacting system is both *deterministic* and *continuous*. In reality, the molecules react

in *stochastic* manner as they collide, and change their concentrations by the *discrete* amounts. This assertion describes exactly the stochastic formulation. The second one revolves around the computational complexity of the algorithm. While 30 years ago, Gillespie considered the computation time to be the biggest limitation of this approach, current processing power overcomes this constraint. A better physical basis, in addition to the available resources, justifies stochastic simulations as a promising technique. In addition, one serious shortcoming of ODE models is that they are unable to describe the concentration *fluctuations* resulting from the stochastic nature of intercellular interactions. The stochastic nature inherent in the probabilistic methods provides addition rationale to this approach. In section 3.2, I provide Gillespie's timed interpretation of biochemical process that is based on the assumption of the stochastic molecular dynamics. *Stochastic Petri Nets* (SPN) take advantage of the discrete state-space modeling approach by dealing with evolution of the biological system at the molecular level.

3.1 Deterministic simulations

In section 2.1 we showed how to derive a set of ordinary differential equations to represent a biochemical system using classical principals of chemical reaction kinetics. Looking back at the system 2.8, a conventional abbreviation is to write such and ODE in the form:

$$\begin{aligned}
\frac{d[S]}{dt} &= f_S([E],[S],[ES]),\\
\frac{d[E]}{dt} &= f_E([E],[S],[ES]),\\
\frac{d[ES]}{dt} &= f_{ES}([E],[S],[ES]),\\
\frac{d[P]}{dt} &= f_P([ES]).
\end{aligned} \tag{3.1}$$

with

$$\begin{aligned}
f_S([E],[S],[ES]) &=_{def} -k_1[E][S] + k_{-1}[ES],\\
f_E([E],[S],[ES]) &=_{def} -k_1[E][S] + k_{-1}[ES] + k_2[ES],\\
f_{ES}([E],[S],[ES]) &=_{def} +k_1[E][S] - k_{-1}[ES] - k_2[ES],\\
f_P([ES]) &=_{def} +k_2[ES].
\end{aligned} \tag{3.2}$$

Introducing further a concentration vector $\overrightarrow{x}$ with components $x_1 = [E]$, $x_2 = [S]$, etc., as well as a rate vector $\overrightarrow{f}(\overrightarrow{x})$ with components $f_1(\overrightarrow{x}) = f_S([E],[S],[ES])$, $f_2(\overrightarrow{x}) = f_E([E],[S],[ES])$, etc. we yield the general form

$$\frac{d\overrightarrow{x}}{dt} = \overrightarrow{f}(\overrightarrow{x}). \tag{3.3}$$

Solving this equation gives us concentration of all species at time t. Additionally, by setting $\overrightarrow{f}(\overrightarrow{x}) = 0$, we can perform steady-state analysis and define values of the concentration at which their time derivation becomes zero.

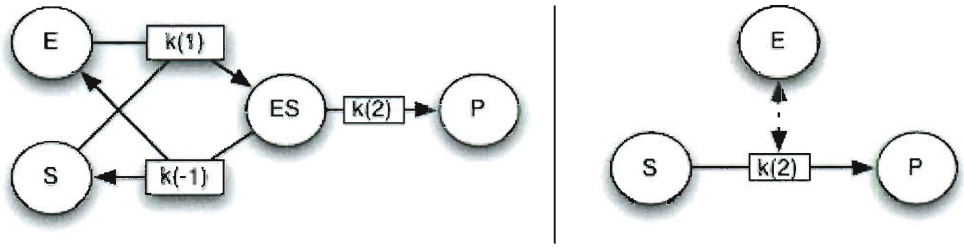


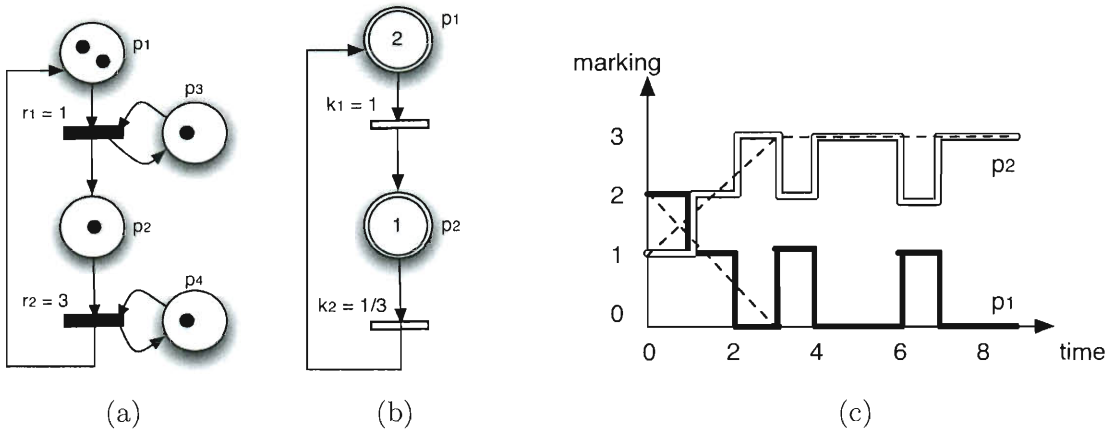
Figure 3.1: Example of converting ODEs to PNs. The left panel shows the explicit conversion of an ODE set from the equation refeq:xdef to the PN structure. The right panel shows a simplified Petri net with one enzymic reaction, which assumes that the compound is formed and the rate is simplified to k_2.

As mentioned earlier in section 2.1 any set of ODEs can be uniquely defined by the Petri net structure. Figure 3.1 shows a Petri net representation of the ODEs in Equation 2.8.

3.1.1 Continuous Petri Nets

Continuous Petri nets (CPN) can be considered a graphical representation of the systems of ordinary differential equations. CPNs are derived from discrete ones by assigning rate equations to all atomic actions, and thus, is nothing else than a structured description of ODEs. See Figure 3.2 for an example of converting TPN to CPN.

In a CPN the marking of a place is no longer an integer, but a positive real number, called token value (which is interpreted as a concentration of a given species). The instantaneous firing of a transition is carried out like a continuous flow, whereby the current firing rate depends generally on the current marking. A transition is



(a) (b) (c)

Figure 3.2: From discrete TPN to CPN: (a) Discrete timed PN, where places p_3 and p_4 correspond to *explicit limitations* of transitions t_1 and t_2 respectively. Rates r_1 and r_2 represent the firing times of the corresponding transitions, meaning the number of tokens needed to fire. (b) Conversion to CPN, where k_i represents the *maximal firing speed* of a transition (and corresponds to kinetic constants of ODEs). (c) The behavior of both models in terms of concentrations of p_1 and p_2. Discrete models are shown in filled (p_1) and open (p_2) solid lines. Dashed lines show continuous concentrations in CPN.

considered enabled if the continuous marking is greater then zero in all the pre-places. Due to the influence of time, a continuously enabled transition is forced to fire as soon as possible. In CPN, each transition has weight, k_i, representing kinetic parameters of ODEs. The firing rate of a transition depend on the weighted concentrations of the substances involved:

$$r(t_i) = k_i \times \prod_{p_j \in {}^\bullet t} m(p_j) \tag{3.4}$$

where ${}^\bullet t$ are activation pre-places of transition t.

Altogether, the semantic of CPN is defined by a system of ODE, where one equation describes the continuous change over time on the token value of a given place

Algorithm 1 Deterministic simulation: CPN

1. Set the initial number of tokens and set time $\tau = 0$
2. Determine the set of enabled transitions $E(m)$.
3. For each enabled transition $t_i \in E(m)$, calculate $r(t_i)$ using Equation 3.4.
4. For each place p calculate $\Delta m(p)$ using Equation 3.5.
5. For each place p update its marking $m(p) = m(p) + \Delta m(p)$.
6. Set $\tau = \tau + 1$. Go to step 2.

by the continuous increase of its pre-transition flow and the continuous decrease of post-transition flow:

$$\Delta(m(p)) \equiv \frac{m(p)}{dt} = \sum_{t \in {}^\bullet p} r(t) \times insc(t,p) - \sum_{t \in p^\bullet} r(t) \times insc(p,t). \tag{3.5}$$

where ${}^\bullet p$ and $p^\bullet$ respective incoming and outgoing transitions of place p, both weighted by the inscriptions of the appropriate arcs. Essentially, the CPN becomes the structured description of the corresponding ODEs. Due to explicit structure we expect to get descriptions which are less error prone compared to those ones created manually from scratch [33]. Full algorithm for simulating a CPN is shown in Algorithm 1. The algorithm terminates when the τ reaches some pre-defined max value.

3.2 Stochastic simulations

The history of the stochastic simulations began in 1976 when Gillespie suggested an algorithm, which uses rigorously derived Monte Carlo procedure, to numerically simulate the time evolution of a given chemical system [2]. Generally, in modeling of biochemical systems it is necessary to reflect the stochastic nature of the systems due to cell to cell variation, or to describe external noise (generated by fluctuations of the environment), or intrinsic noise (due to low molecular concentrations). Gillespie's *stochastic simulation algorithm* (SSA) is a numerical simulation procedure that is essentially exact for spatially homogeneous or well stirred chemical systems. SSA is considered exact because it is rigorously based on the same microphysical principles that underly the *chemical master equation* (CME) [34].

The Chemical master equation is a traditional method of calculating the stochastic time evolution of a chemically reacting system [35]. To derive CME, we consider a system that involves N species $\{S_1, \ldots, S_N\}$ where the state of the system at time t is represented by the vector $X(t) = (X_1(t), \ldots, X_N(t))$, where $X_i(t)$ is the number of molecules of species S_i at time t. The system consists of M reactions $\{R_1, \ldots, R_M\}$. The dynamics of each reaction R_μ is characterized by the *propensity function* a_μ. The approach leads rigorously to the following statement: the probability that a certain reaction μ will take place in the next instant of time $\mathrm{d}t$ is driven by $a_\mu \mathrm{d}t + o(\mathrm{d}t)$, where a_μ is independent of $\mathrm{d}t$, and $o(\mathrm{d}t)$ denotes terms that are negligible for small $\mathrm{d}t$. However, a_μ may depend on (a) the reaction rate, (b) the current number of molecules involved in the reaction, and (c) the current time. Hence the propensity

of the reaction R_μ is defined by

$$a_\mu = k_\mu \times \prod_{p_i \in {}^\bullet t} m(p_i) \times \mathrm{d}t + o(\mathrm{d}t) \tag{3.6}$$

where k_μ is a reaction rate, ${}^\bullet t$ are activation pre-places of transition t, and $m(p_i)$ is a marking of place p_i. Given this stochastic framework and the system in the state $X(t)$, the probability of the reaction R_μ executing and evolving the system into $X'(t)$ is defined by its propensity

$$P(X'(t), t + \mathrm{d}t | X(t), t) = a_\mu \mathrm{d}t + o(\mathrm{d}t). \tag{3.7}$$

Note that because the transition probability depends only on the current state and not on the previous states, the underlying process is Markovian.

Given the propensity function $a_\mu \mathrm{d}t + o(\mathrm{d}t)$, the *state change vector* $\boldsymbol{v_\mu} = (v_{1_\mu}, \ldots, v_{N_\mu})$, and the system states $X(t) = x$ (current) and $X(0) = x_0$ (initial), the dynamics of the system obeys the CME:

$$\frac{\partial P(x, t|x_0, t_0)}{\partial t} = \sum_{j=1}^{M} [a_j(x - v_j) P(x - v_j, t|x_0, t_0) - a_j(x) P(x, t|x_0, t_0)], \tag{3.8}$$

where function $P(x, t|x_0, t_0)$ denotes the probability of reaching state $x = X(t)$ given initial state $x_0 = X(t_0)$ [36]. The CME entails writing a system of equations and solving simultaneously for the probabilities of *all* trajectories, therefore it is hard to solve both theoretically and numerically except for very simple systems. A better way to generate evolutions of species is to pick reactions and times according to the correct

probability distributions so that the probability of generating a given trajectory with the simulaiton algorithm is exactly the probability that would come out the solution of the CME. Gillespie's exact stochastic simulation algorithm specifies how to generate random numbers so that they have this correct distribution.

3.2.1 Gillespie's methods

Gillespie proposed two simulation methods [32]. The first method - *direct method* - calculates explicitly which reaction occurs next and when it occurs. The second one - *first reaction method* - generates for each reaction μ a putative time τ_μ at which reaction μ occurs, and chooses the reaction with smallest τ_μ (the 'first' reaction) to be executed.

The Direct method aims to determine two values: *which* reaction occurs next and *when* it occurs. This can be determined by specifying probability density function $P(\mu, \tau)$ of reaction μ executing at time τ. In [2], Gillespie showed that

$$P(\mu, \tau)d\tau = a_\mu \exp(-\tau \sum_j a_j)d\tau. \tag{3.9}$$

where a_μ are reaction propensities. Integrating $P(\mu, \tau)$ over all τ from 0 to ∞ we can calculate the probability of reaction μ^* to be executed:

$$P(\mu = \mu^*) = \frac{a_{\mu^*}}{\sum_j a_j}. \tag{3.10}$$

To determine next time interval τ we sum $P(\mu, \tau)$ over all μ

Algorithm 2 Gillespie SSA: Direct method

1. Set the initial number of tokens, and set time $\tau = 0$
2. Get a set of enabled transitions E.
3. Calculate propensities a_i for all transitions $t_i \in E$ using Equation 3.6.
4. Generate a random number r_μ to represent the probability of the next reaction being μ^*, or $P(\mu = \mu^*)$
5. Using Equation 3.10 determine the reaction μ^* to be executed next with the probability r_μ.
6. Generate random number r_{τ_μ} to represent the probability of the next time interval being τ_{μ^*}, or $P(\tau_\mu)\mathrm{d}\tau$
7. Using Equation 3.11 calculate next time interval τ_{μ^*} with probability r_{τ_μ}.
8. Execute transition μ^* by changing the number of tokens accordingly. Update the time to $\tau + \tau_{\mu^*}$.
9. Go to step 2.

$$P(\tau)d\tau = (\sum_j a_j)\exp(-\tau \sum_j a_j)d\tau. \tag{3.11}$$

Algorithm 2 shows the implementation of the direct method within the Petri net framework.

The First reaction method determines the reaction μ to be executed by assigning putative time τ to all reactions - the time the reaction would occur if no other reaction occurs first. μ^* to be executed will be the reaction with the smallest putative time τ_{μ^*}. Algorithm 3 gives an outline of the process.

Algorithm 3 Gillespie SSA: First reaction method

1. Set the initial number of tokens, and set time $\tau = 0$
2. Get a set of enabled transitions E.
3. Calculate propensities a_i for all transitions $t_i \in E$ using Equation 3.6.
4. For each transition $t_i \in E$
   (a) Generate random number $r_{\tau_i} = P(\tau_i)\mathrm{d}\tau$
   (b) Using Equation 3.11, calculate delay interval τ_i with probability r_{τ_i}.
5. Pick the next time interval τ_{μ^*} to be the smallest delay, and the next reaction μ^* to be the one with this delay.
6. Execute transition μ^* by changing the number of tokens accordingly. Update the time to $\tau + \tau_{\mu^*}$.
7. Go to step 2.

Although the two algorithms seem very different, the probability distributions used to choose μ and τ are the same. In regard to efficiency, it is important to point out that in the first reaction method, we need to generate a random number for every transition in E, which may change with every iteration, where as in direct method we only need two random number at each iteration. Therefore direct method is a little more efficient and is used as a representation of the stochastic strategy in the rest of this work.

In 1997 McAdams *et al.* [37] used SSA in transcriptional network to capture the patterns of signal protein production that determine switching delays in gene expression. The SSA approaches appeared to be extremely successful which led to

them being applied to the larger systems. In 1999, Arkin *et al.* [38] used SSA to simulate a model of simple virus, lambda phage, containing 75 equations and 57 chemical species. However, as this experiment showed, one of the main limitations of the original algorithm is scalability. In the early 2000s many variations of SSA were proposed, mainly focusing on improving the performance and scalability of the algorithm.

3.2.2 Other stochastic simulators

The first successful adoption of SSA appeared in 1999 when Gibson and Bruck proposed *Next reaction method* [39]. The new algorithm utilized *first reaction method* and gained speed by avoiding generation of unnecessary random numbers: storing some parameters to avoid re-calculations, and using more efficient data structures. Similarly, in 2004 Cao *et al.* [34] suggested optimizations to improve upon the original *direct method.* In addition to using similar techniques as Gibson and Bruck, he suggested the ordering for reactions based on the frequency of their execution to improve the efficiency of drawing from exponential distribution.

In 2001 Gillespie proposed a new *approximate* SSA solution [40]. To address the lack of computational efficiency of the original SSA, he proposed the "τ-leap" method to produce significant gains in the simulation speed with acceptable losses in accuracy. This approach is based on the fact that each time series evolution can be divided into a set of continuous subinterval in such a way that, if we could determine only how many times each reaction fired in each subinterval, we could omit knowing the precise instants at which those firings took place. That would allow us to *leap* through the

system's evolution by some pre-selected interval τ and achieve computational gains. In order to maintain the acceptable level of accuracy it is important to pick τ that satisfies a so-called *leap condition*. The leap condition requires τ to be small enough that the change in the state during $[t, t+\tau)$ will be so slight that no propensity function will suffer a significant change in its value [40]. In practice that means that the absolute fractional change in the propensity a_j of each reaction R_j during the time interval τ, will not exceed some sufficiently small ϵ, given by:

$$\left|\Delta_\tau \frac{a_j(X(t))}{a_j(X(t+\tau))}\right| < \epsilon, \tag{3.12}$$

where $X(t)$ is a state vector at time t. Once τ is determined, one needs to figure out how many times each reactions occurs in this time interval. The key to doing it properly lies in the *Poisson random variable*:

$$k_j = \mathcal{P}(a_j, \tau), \tag{3.13}$$

where k_j is the number of times the reaction R_j fires in time interval $[t, t+\tau)$ given its propensity a_j. By assigning k_j to each reaction, we can get the net change of the system defined by the vector:

$$\boldsymbol{\lambda} = \sum_{j=1}^{M} k_j \boldsymbol{v_j}, \tag{3.14}$$

where $\boldsymbol{v_j}$ is a state change vector after k_j instances of the reaction R_j have occurred. At each leap, the state of the system is adjusted by $\boldsymbol{\lambda}$ [41].

Although, this method showed significant speed up improvements, it suffered one

serious shortcoming due to the unbounded nature of the Poisson distribution. Under some conditions, especially when the molecular populations are small and/or the leap condition is large, the method could produce physically unrealistic negative population sizes of individual states. To address this problem, Tian *et al.* [42] and Chatterjee *et al.* [43] independently developed binomial τ-leap method. This method used binomial distribution to determine the number of firings in the time interval τ. The distribution's bounded nature ensured mass conservation of the system. Numerical comparisons with the original τ-leap method showed good accuracy and significant efficiency.

However, in 2005, Cao *et al.* suggested a few major improvements to the original explicit Poisson τ-leap method making it again comparable to the binomial version. First, they considered distinguishing critical and noncritical reactions in a chemically reacting system in order to resolve negative populations issues [44]. Upon successfully validating the results, they proceeded with further modification to improve on the step size selection technique by using relative changes of the propensity functions and reactant species [45]. These modification showed to be most significant in the efficiency gains for the systems with large number of reactions.

In the history of stochastic simulation algorithms, the exact original implementation provided high level of accuracy. Later race for SSA efficiency focused on its applicability to larger networks while sacrificing some accuracy. In the next section, I examine the accuracy of selected algorithms.

3.3 Example and comparisons

In order to verify to performance of my Petri net simulator, I use the model and the data produced by Vayttaden *et al.* in their work on the $\beta 2$ adrenergic receptor ($\beta 2$AR). $\beta 2$AR is intimately involved in the control of smooth muscle relaxation in airways and the vasculature, in stimulation of the heart, and numerous other physiologically important actions [46]. In this study they curate a model of G protein coupled receptor kinase (GRK) regulation of the $\beta 2$AR pathway, and examine the dynamics of the $\beta 2$AR responses to different concentrations of the the drug. In Figure 3.3 (a) I show the reaction description of the networks and on Figure 3.3 (a) its Petri net representation.

The responses of interest include (1) GSK phosphorylation and dephosphorylation denoted by R^* and R_g respectively; (2) surfacing and internalization of $\beta 2$AR denoted by R_s and R_i respectively; (3) $\beta 2$AR desensitization and resensitization, measured by comparing the active forms of $\beta 2$AR. The ODE-based mathematical approach is used to extract the dynamics of the system. The parameters for the system were experimentally determined with rate $k_{2f} = 1.4\alpha[R^*]$ being the response to a varied agonist concentration (where α is a parameter relevant to the agonist coupling efficiency). The results of the simulations are, then, compared with the *in vitro* experiments. In Figure 3.4 we show the evaluation of deterministic model in [46] against their experimental data (exp-a, exp-b). For the different concentrations of the treatment, we obtain the curves for the total amount of phosphorylated receptors (exp-a) and the total amount of surface receptors (exp-b). On the bottom two panel, I show the re-

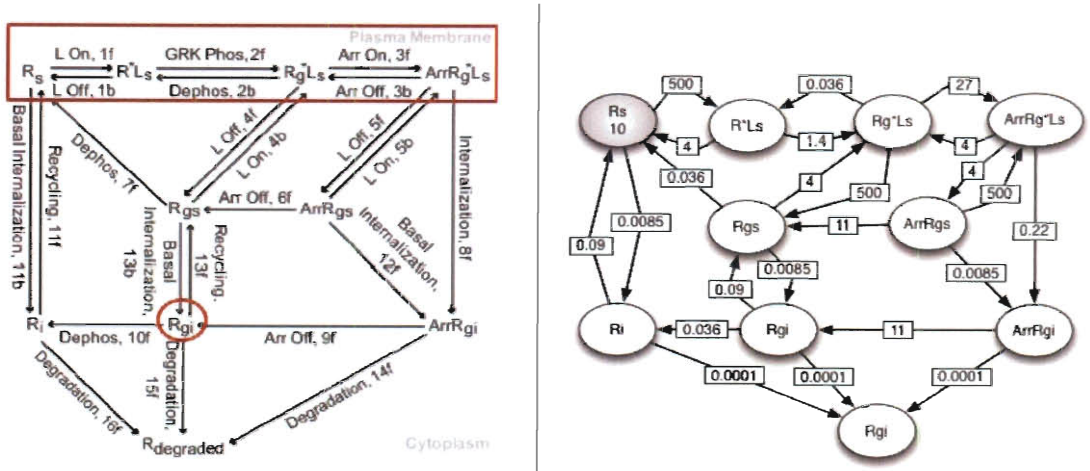


Figure 3.3: Reaction diagram and the Petri net representing the GRK-mediated β2AR regulation: (a) L is ligand; R^* is active state of β2AR; R_s and R_i are surface/plasma membrane and internalized β2AR; R_g is GRK-phosphorylated β2AR; Arr is arrestin. Adopted from [46]. Species contained in red square (surface behavior) and circle (internalization) are analyzed in detail in section 3.3. (b) A Petri net representing this system. Each reversible reaction is split into two transitions. The protein Rs shown in grey is the only one that has initial concentration (set to 10 tokens).

sults of my Petri net based simulator against the data from [46] (sim-a, sim-b). Both *direct method* stochastic (dashed lines) and continuous (solid lines) simulators seem to produces the exact trends as the experimental data. This provides the evidence for the validity of Petri net simulator executions.

To further address the accuracy of PN based simulator, we use β2AR system to separately analyze the behavior of each species during the simlatoin. Both deterministic (CPN) and stochastic (SPN) time courses produced by our PN-based simulator are compared with COPASI [47] simulations. In Figure 3.5, we show the comparison of our Petri net simulator to the results acquired in COPASI. The simulated system is

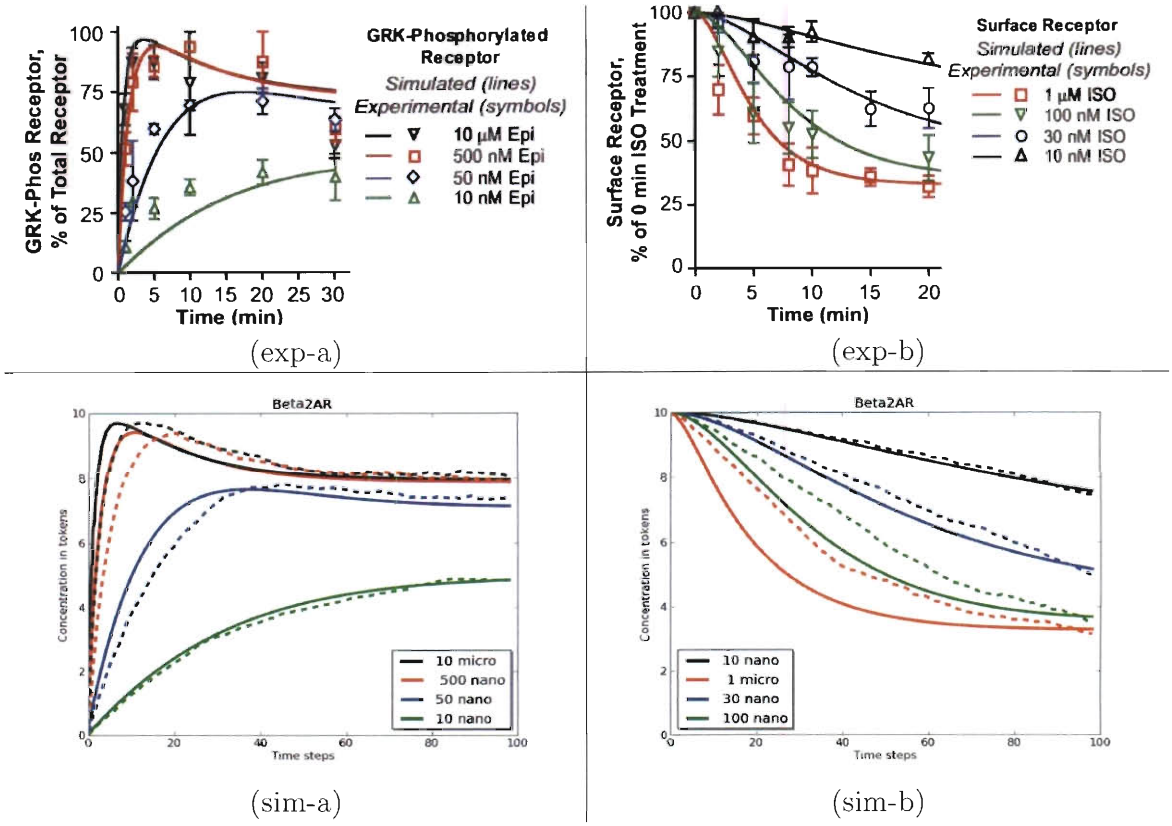


Figure 3.4: Validation of simulations against the experimental data in [46] : [Top] The points are experimental data. The lines resulted from the ODE model curated in [46]. (exp-a) Time course of GRK phosphorylation of the receptor on treatment with different concentrations of of the treatment. (exp-b) Internalization of the β2AR on treatment with various concentrations. [Bottom] Gillespie's direct method (SPN) and CPN simulations. 10 tokens is used for a single point of entry at Rs. (sim-a) Verifies my Petri net simulation against the data in (exp-a). Plotting the sum of all phosphorylated species (the ones with the 'g' in the name). (sim-b) Verifies my Petri net simulator against the data in (exp-b). Plotting the sum of all the surface proteins (the ones in the red box in Figure 3.3(a)). In (sim-a) and (sim-b) solid lines represent continuous simulator and dashed lines correspond to the results of the *direct* method.

β2AR network with 10 μmol treatment. For deterministic solution, we compare the outcome of CPN simulator versus LSODA [48] ODE solver in COPASI. The results

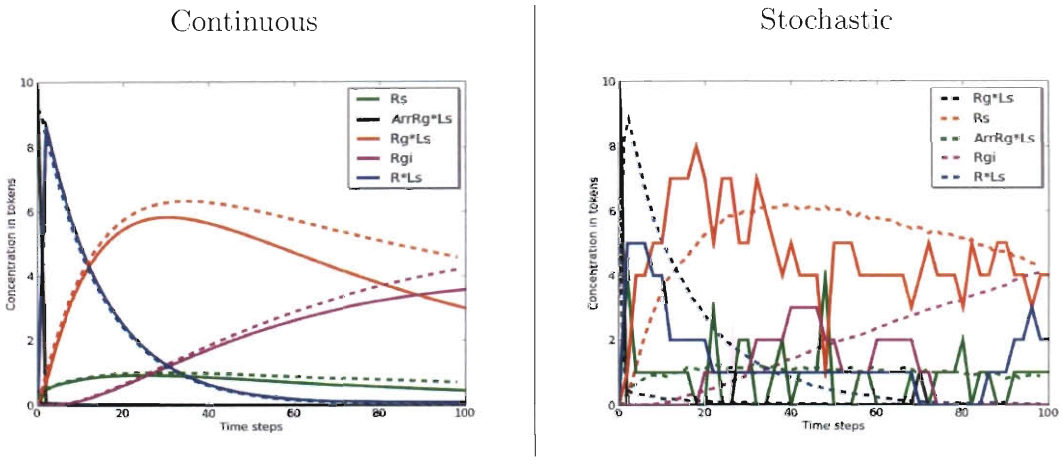


Figure 3.5: Petri nets vs COPASI: comparing on β2AR system with 10 μmol treatment. COPASI simulations were done using the exact model constructed in [46]. The model was constructed in the dimensionless volume with the unit quantities measured in '#' (COPASI unit), which 10 set as initial concentration of *Rs*. On the left panel, CPN simulation (dashed lines) is compared with COPASI ODE solver (solid lines). On the right panel, my Petri net implementation of Gillespie's *direct method* (dashed lines) is compared with COPASI's implementation of *direct method*

show exact same trends with small differences in the amounts. For stochastic validation, I compare my Gillesie's *direct method* implemented in my work to COPASI's implementation. While we see similar overall trend between the two, the dashed lines representing implementation from my tool are smooth, whereas solid lines from COPASI's implementation are rather choppy. This is because they represent *one* iteration of stochastic simulation. Generally, in order to get smoother curves and more statistically significant results for stochastic simulations, SPN instances has to averaged over multiple runs (which is the case for dashed lines). In the next section I discuss the particularities of SPN instances and appropriate methods for merging them.

3.4 Aligning Gillespie instances

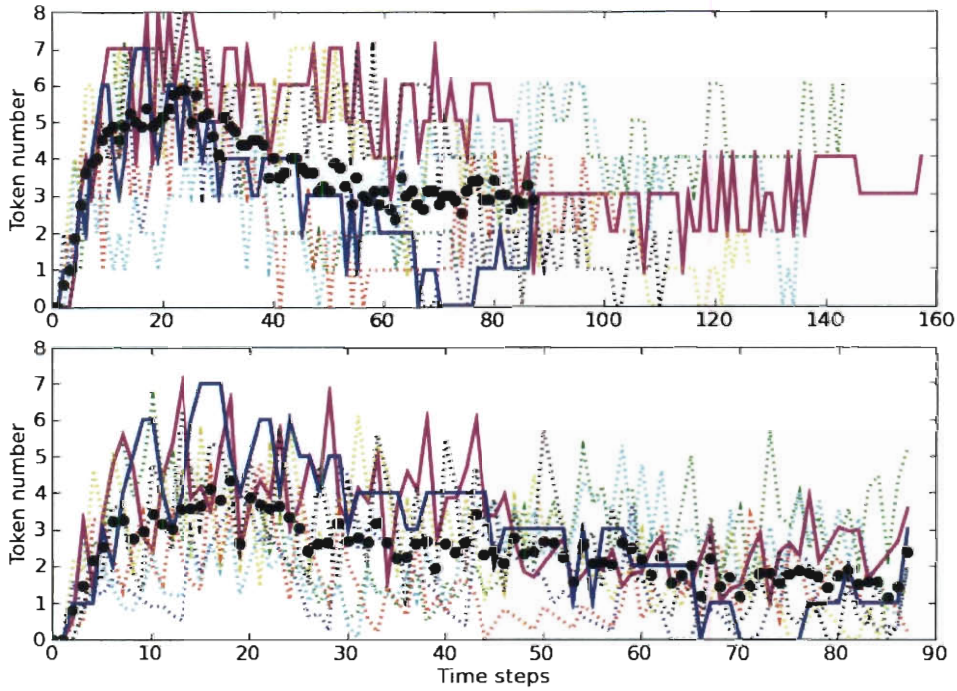


Figure 3.6: Averaging SPN instances of $ArrRg^*Ls$ in β2AR system. [Top] Raw individual time series. All instances are of different length due to the stochastic nature of SPN execution. Solid lines show the shortest and the longest executions. [Bottom] Time series after applying PAA reduction to the time series. Solid lines show the shortest and the longest instance. Black dots show the average of the instances in each case.

In Gillespie algorithms, the time step selection is based on the random variable, and calculated according to Equation 3.11. This means that every run of SPN would result in the time series courses of different lengths. In Figure 3.6, we examine the time course of compound $ArrRg^*Ls$ in the β2AR system. The top image shows the eight raw instances of SPN execution. Thicker, solid lines show the shortest and the

longest instance. These instance are of varying lengths due to the stochastic nature of SPNs. The black dots show what would be the average of this species if we did element by element averaging up to the shortest sequence. The bottom plot shows the instances after they were aligned while preserving the general structure of the series. The black dots show the average of these series. In the raw data, all instances exhibit the similar behavior: we see an increase in concentration followed by its decline. This feature needs to be accurately captured while averaging. Averaging the raw data up to the shortest instance looses a lot of information in the tails of the rest, and biasing the results by combing discordant pieces. In order to correctly extract the features of all the times series, we apply the Piecewise Aggregate Approximation (PAA) algorithm proposed by Keogh *et al.* [49]. In this algorithm, we convert the series of different length into the length of the shortest one preserving their structural features. Details of PAA algorithm are discussed in section 3.4.1. Bottom plot in Figure 3.6 shows the PAA conversion of all eight time series with the solid lines (former shortest and longest series) exhibiting the same features. The black dots are the average of these aligned time series, which more accurately captures the overall behavior. PAA conversion preserves the features of the time series and is a necessary step before any averaging or combining of SPN instances.

3.4.1 Piecewise Aggregate Approximation (PAA)

PAA algorithm achieves alignment of the time series data while preserving their structural characteristics. The algorithm originated to solve the problem of similarity search in large time series databases. The method is motivated by the simple obser-

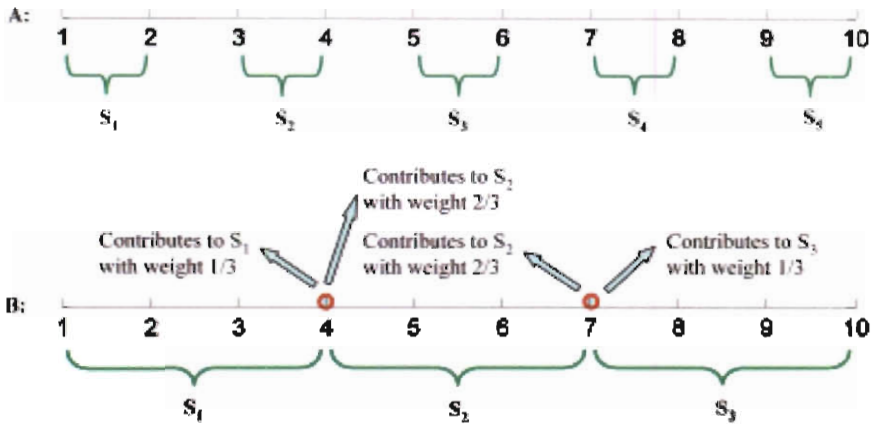


Figure 3.7: PAA example: A) 10 data points are divided into 5 segments. B) 10 data points are divided into 3 segments. The data points marked with circles contribute to two adjacent segments at the same time. *Li Wei and Eamonn Keogh, 2006 SAX Manual*

vation that for most time series datasets we can approximate the data by segmenting the sequences into equi-length sections and recording the mean value of these sections [49].

Given a time series of length m, we want to reduce it to the length n, where $n < m$, while preserving its structural characteristics. We apply PAA to the time series by dividing it into n pieces and recording the means in these intervals. If m is not divisible by n, there will be some points in the time series that would partially fall into two intervals. Figure 3.7 shows an example of PAA division. In Figure 3.7(A), we are dividing 10 data points into 5 segments, resulting in integer division where two full points belong to each interval. In Figure 3.7(B), we are dividing 10 data points into 3 segments. Since 10 is not divisible by 3, it is not clear where points 4 and 7 belong. To guarantee equi-lenght division, instead of putting the whole point into a segment, each boundary point can partially contribute to both segments. For

example, since $10 \div 3 = 3\frac{1}{3}$, we expect point 4 to contribute $\frac{1}{3}$ of its value to segment 1 and the rest to segment 2. Now segment 2 already has $\frac{2}{3}$ out of $3\frac{1}{3}$ needed values, therefore, only $2\frac{2}{3}$ more points needed. So the segment 2 will also include points 5, 6 and $\frac{2}{3}$ of point 7. As this continues, each segment will contain $3\frac{1}{3}$ points. For each segment, included values are added and divided by the segment size to find the mean, in this example by $3\frac{1}{3}$. After PAA is applied, we can adopt many different distance measures to compare resulting time series of the same length.

In this chapter we discussed simulation strategies that will be tested for the robusteness. We validated their performance on experimental data and against other simulation implementations. We addressed particularities of averaging SPN instances to produces smooth curves. In the next chapter we discuss the analysis setup for our investigation on error-robustness of Petri net in signaling network.

Chapter 4

Analysis setup

In this chapter I discuss the methods for testing the robustness of the Petri nets simulators in signaling networks. First, I introduce the biochemical systems used for testing. Then I describe the procedure for systematically introducing the error into the system. Lastly, I expand on the the methods for measuring the distance between the error-free time series and the ones of the perturbed system.

4.1 The data

I evaluate the robustness of the Petri net simulators on two networks. The first one is the $\beta 2$ adrenergic receptor ($\beta 2$AR) introduced in Section 3.3. This system is chosen because it is relatively small in the number of nodes but is highly interconnected as it has many reversible reactions (see Figure 3.3).

The second system is based on the signaling networks and the system of ODEs

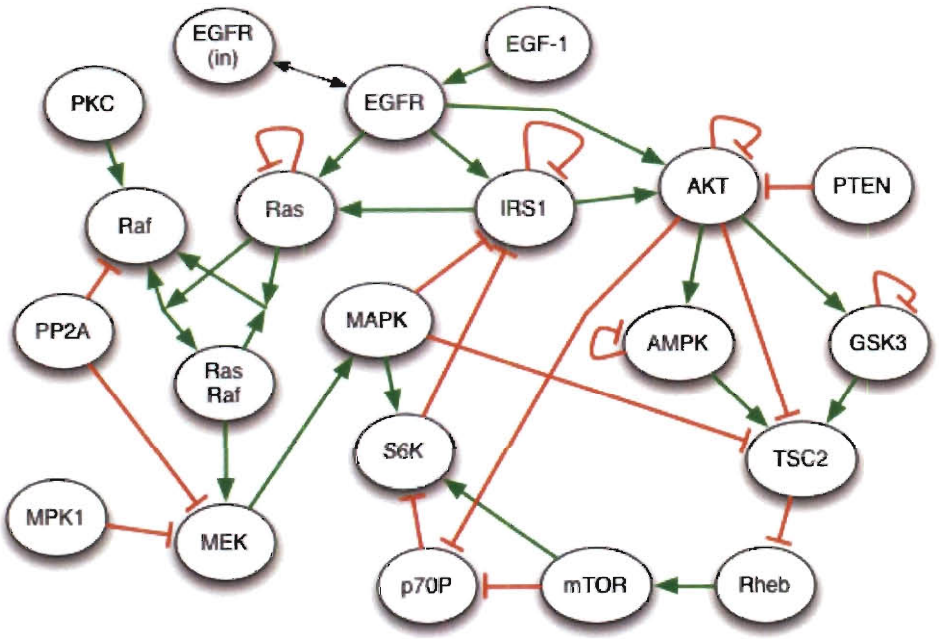
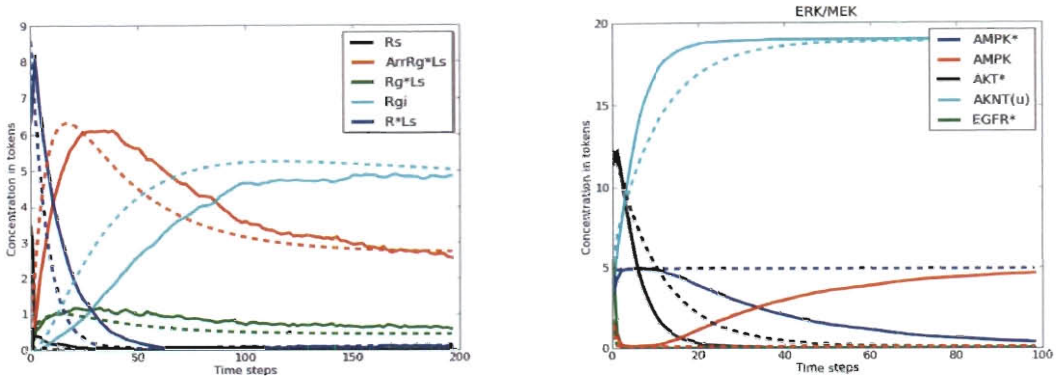


Figure 4.1: Reaction diagram of EGFR network. Arrows represent activation of the protein to its phosphorylated form. Plungers indicated deactivation of an active form of the protein. In this system, I model $EGFR$ simulation, meaning that there is elevated number of tokens present at the entry point in EGF-1.

given in [50]. The topology of the underlying network is shown in Figure 4.1. Additionally, they used reverse-phase protein array (RPPA) to measure the transient response of the cell line after stimulation by insulin-like growth factor (EGF-1). In [50] Iadevaia *et al.* used this time series data to parametrize the set of ODEs for this cell line. This system is particularly good as it includes many enzymic activations and inhibitions, several competing reactions, and circular feedback interactions. I use the parameter set and the initial concntrations acquired by the *Particle Swarm Optimization* (PSO) algorithm in [50] as a starting point for the error robustness checking. In



(a) (b)

Figure 4.2: Simulations of the original networks: (a) β2AR system with error-free kinetic parameters and initial concentrations. Modeling the propagation of the treatment from *Rs* to internalization *Rgi*. (b) Erk/Mek pathway with error-free kinetic parameters and initial concentrations. *Solid* line is stochastic simulation and *dashed* line is deterministic simulation. Modeling $EGFR$ stimulation.

this system, I model the $EGFR$ simulation, meaning that there is elevated number of tokens present at the entry point in EGF-1.

For my setup, I have two biochemical systems with a known and complete topology[1], a complete set of kinetic parameters and the necessary initial concentrations. In the next section, I discuss the ways to systematically perturb the system to introduce the desired amount of uncertainty. In Figure 4.2, I show the simulation outcomes given the error-free systems. Stochastic SPN and deterministic CPN simulations are almost identical for β2AR system. However, in the case of Erk/Mek network we see an interesting behavior in the forms of $AMPK$ protein. Particularly, in the SPN simulation we see phosphorylated state ($AMPK^*$) and inactive state ($AMPK$)

[1]For the purpose of this study we consider both topologies complete.

	Description	Reaction	Rate
1	Self-activation	$AKT(u) \rightarrow AKT^*$	0.002
2	Phosphorylation	$AKT + EGFR^* \rightarrow AKT^* + EGFR^*$	5.66
3	Self-inhibition	$AKT^* \rightarrow AKT$	0.29
4	Deactivation	$AKT^* + PTEN \rightarrow AKT(u) + PTEN$	9.99
5	Phosphorylation	$AMPK + AKT^* \rightarrow AMPK^* + AKT^*$	4.28
6	Self-inhibition	$AMPK^* \rightarrow AMPK$	0.07

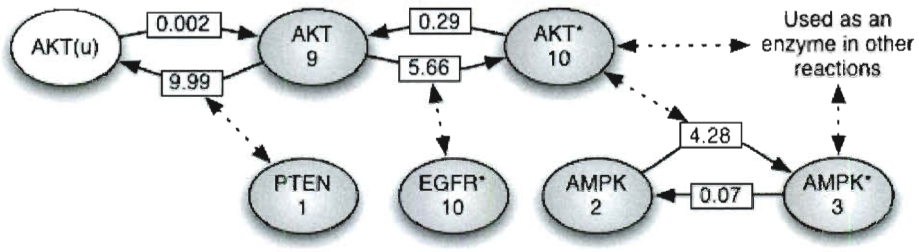


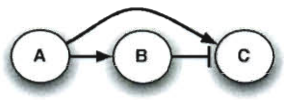
Figure 4.3: Immediate dynamics of AMPK in Erk/Mek model: Top table shows the reactions that immediately affect the dynamics of $AMPK$. Bottom image show the Petri net representation of above interactions. Solid edges represent the token (mass) transfer; dashed edges represent enzymic influence (concentration does not change, but some amount needed to make corresponding transitions enabled). Colored nodes have initial marking as noted on the illustation. We see that $AMPK$ gets phosphorylated by AKT^*, which has a strong influence of $PTEN$ towards deactivation.

reaching opposite steady states. In Figure 4.3, I show the exact reactions and the Petri net network corresponding to the dynamics that immediately affects $AMPK$ activity. From this figure, we see that AKT^* is the only enzyme that effects the balance of $AMPK$ mass. While the AKT^*'s phosphorylation rate on $AMPK$ is sufficiently high (reaction 5), there is even stronger force of $PTEN$, that is driving APK towards de-actionvation (reaction 4). $PTEN$ is an ezyme that stays always present in the system and eventually drives AKT towards fully inactive state (black line in Figure 4.3). Once there is no more AKT^* left in the system (that acts as an

enzyme for $AMPK$), we expect $AMPK$ to move towards de-phosphorylation. This is clearly captured in the SPN simulation (blue and red lines in Figure 4.3(b)), but not reflected on the CPN simulation. This supports the physical bases that Gillespie accounted for in the stochastic approach: "deterministic simulations assume that the time evolution of a chemically reacting system is both *deterministic* and *continuous*; in reality, the molecules react in a *stochastic* manner as they collide, and change their concentrations by the *discrete* amounts." When AKT^* becomes exhausted, in SPN it will actually have zero-concentration, therefore, disabling reaction 5. Whereas in CPN, its zero-concentration is a limit going to zero, therefore, reaction 5 stays enabled, but changes the concentrations of $AMPK$ by the amounts that are almost negligible and it appear to stay constant. This behavior also manifests itself when the sub-unit directly affecting $AMPK$ was modeled in isolation (see Figure A.1 in supplementary material).

4.2 System perturbations

One of the central goals of this work is to determine how the uncertainties in the original model description affect the performance of the Petri net simulators, both continuous and stochastic. In order to achieve this goal I devise a method to introduce error into the system in the controlled manner. I systematically perturb three components of each system: (1) kinetic parameters, (2) initial values, and (3) topology. Petri net simulations are based on the law of mass action kinetics and are highly dependent on the accurate estimates of each of the three components. The main two



(a)

Reaction	Kinetic Param
A → B	100
A → C	0.1
B −| C	10

(b)

Protein	Initial value
A	10
B	0
C	10

(c)

Figure 4.4: "Toy" system used in explaining testing procedure: We assume the system has persistent production of A (meaning A is constantly produced at some rate to avoid system exhaustion) and the sink of excessive B with same (low) rate. (a) An arbitrary "toy" network with competing reactions for C. (b) Shows a set of arbitrary kinetic parameters that consider to be the "true" parameters for this system. With this parameter set we identify the pathways $A \rightarrow B \dashv C$ as dominant. (c) The initial concentrations for this system.

ways of acquiring parameter measures are from the experimental approximation or by deriving them from published experimental studies. However, the values reported in the literature may differ by the orders of magnitude, depending on the experimental conditions. Additionally, experimentally determined parameters come from purified components with no definitive proof that they stay unaltered under other conditions [51]. All these factors account for the small amount of inaccuracy in many biochemical models. Therefore, it is important to acknowledge the threshold when these errors becomes too significant and causes bogus prediction results.

In Figure 4.4 we show a "toy" system to serve as an example to demonstrate the error testing procedure. In the network we have two competing reactions to modify the state of protein C: (1) A activates B which, in turn, de-activates C, or (2) A directly upregulates C. Notice that with the given set of kinetic parameters, we expect the behavior (1) to be dominant as the reaction rate between A and B is much

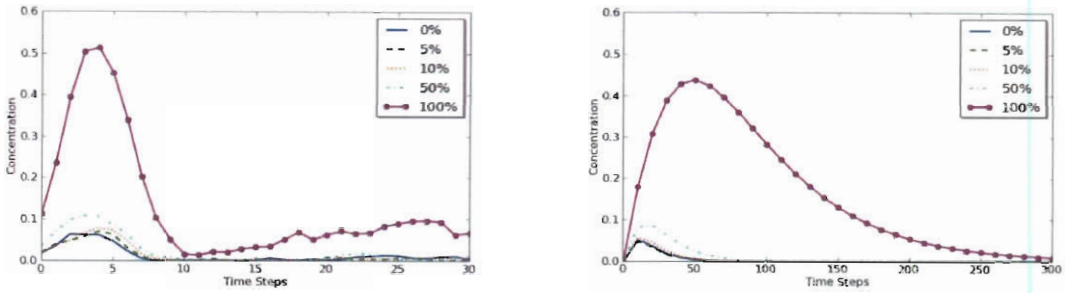
higher than that of A and C. Therefore we expect the protein C to stay inactive.

Kinetic parameters: I randomly modify each parameter within some percent of its original value as the range of allowed change increases. In Table 4.1, I show an example of *one instance* of parameter perturbation run. As I increment the allowed amount of perturbation by some discrete percentage up to the *max* of a 100%, I simulate the system with modified parameter set. For the amount of error "by x%", the random variations are drawn uniformly from the range $[-x, +x]$. The outcomes of the simulation is compared to that of the original network. In Figure 4.5, I illustrate the behavior of protein C (phosphorylated amount of C) under the parameters in Table 4.1. From these plots, we see that, overall, regardless of the parameter error, the concentration of C exhibits a spike in the beginning of simulation, but comes down to the near-zero steady state in the end. This seem to be true for all levels of error shown except 100%. In the case of 100% we see that the protein C not only shows a much higher spike, it also stays in a different steady state. This is due to the fact that this particular instance of 100% error produces very close kinetic parameters for two competing reactions. A stronger activation of C by A is not fully inhibited by B and, therefore, results in the different final state of the system. This makes the alternative path of $A \rightarrow C$ to become dominant.

Initial values: In section 2.2.1, it was mentioned that the concentration unit in Petri net is a *token*. Tokens are represented with integers. In SPN, we move whole number of tokens with each reaction; in CPN we can have decimal values in the transient process. For the purposes of fair comparison, we only allow integer

Original		By 5%		By 10%		By 50%		By 100%	
A → B	100	+ 3%	103	- 9%	91	- 33.6%	66.4	- 82.8%	17.2
A → C	0.1	- 4%	0.096	+ 2.5%	0.1025	+ 24%	0.124	+ 99%	0.199
B −| C	10	+ 0.6%	10.06	- 6%	9.4	- 25.3%	7.47	- 67%	3.3

Table 4.1: Parameter testing example. Showing the percentage of allowed change along with the resulted parameter. Perturbation by $x\%$ refers to the uniform range $[-x\%, +x\%]$.



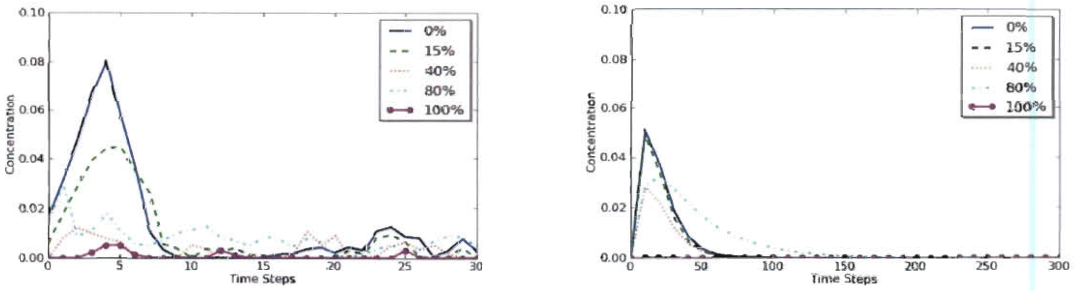
Stochastic simulation Continuous simulation

Figure 4.5: Example for parameters testing procedure: The resulting time series for protein C in "toy" network shown in Figure 4.4 and parameter sets shown in Table 4.1. At 100% inaccuracy, we see that the pathway $A \rightarrow C$ becomes dominant.

values as the initial values of token. The procedure for perturbing initial values is similar to the kinetic parameters, with the only exception that the new value has to be an integer and be > 0. In Table 4.2, I show an example of a initial values perturbation and in Figure 4.6 illustrates the behavior of protein C under the new conditions. While the steady state of C stays approximately the same, the duration and the altitude of the spike varies with different initial concentrations. In addition, in the case of 100%, we see no peak at all, which could be explained by the very low overall concentrations in the system.

Original		By 15%		By 40%		By 80%		By 100%	
A	10	+ 8%	11	- 9%	9	- 73.6%	3	- 99.8%	0
B	0	- 4%	0	+ 22.5%	0	+ 66%	0	+ 99%	0
C	10	- 11.1%	9	- 37.8%	6	+ 55.3%	16	- 67%	3

Table 4.2: Initial values testing example. Showing the percentage of allowed change along with the resulted initial concentrations. Perturbation by $x\%$ refers to the uniform range $[-x\%, +x\%]$.



Stochastic simulation Continuous simulation

Figure 4.6: Example for initial values testing procedure: The resulting time series for protein C in "toy" network shown in Figure 4.4 and initial values sets shown in Table 4.2. At the 100% inaccuracies we see very little activity in the system due to the small amount of the overall mass.

Topology: The network topology incompleteness can manifest itself in two different ways: missing proteins (nodes); missing or incorrectly routed interactions (transitions). In testing simulator robustness to the incompleteness of the network, I separately address both issues by randomly removing nodes (places) and edges (transitions). The testing is also done in the incremental manner, except the *max* allowed value is expressed not in the percentage of error, but in the number of nodes/edges removed. This means, I remove one random node/edge and determine the error, then two random nodes/edges, etc. until reaching pre-defined *max* number.

Original		By 5%		By 10%		By 50%		By 100%	
A → B	100	+ 3%	103	− 9%	91	− 33.6%	66.4	− 82.8%	17.2
A → C	0.1	− 4%	0.096	+ 2.5%	0.1025	+ 24%	0.124	+ 99%	0.199
B −| C	10	+ 0.6%	10.06	− 6%	9.4	− 25.3%	7.47	− 67%	3.3

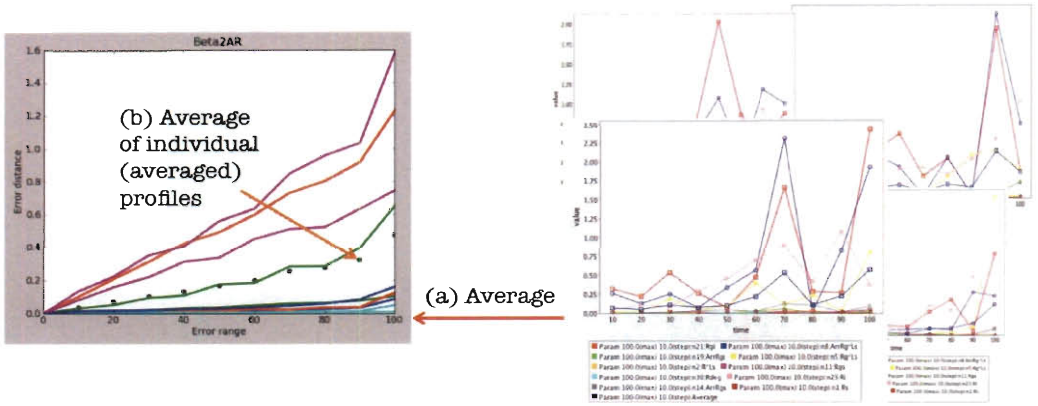


Figure 4.7: Averaging in the error testing procedure:. In the testing procedure, I generated multiple instances of error profiles. In (a) I find the average of those profiles to produce one consensus profile representing errors for individual proteins in the system. To quantify the whole system's response to inaccuracies, in (b) I average all individual error curves in the consensus profile.

I explained the process for systematically introducing the error into the system to determine the robustness of Petri net simulators to the inaccuracies in the model. Since the testing is achieved by introducing random amount of error, in order to quantify the outcomes in the meaningful way it is important to look at the averaged values. In Figure 4.7, I depict the averaging procedure. I generate 200 single error profiles for each system. The consensus profile is then used to find the average error value between all species. In the next section I discuss the distance measures I employ to calculate the error to generate a single error profile.

4.3 Distance measures

In order to compare two sets of simulation results, we are interested in quantifying the difference in some way. I employ two distance measures: *tail distance* and *total distance* between two time-series. The former equates to a biologist examining a single output data (e.g. western blots). The latter equates to making predictions from analyzing time series data. In the next section I describe methods, I have considered to represent both measures and select one way to represent each.

4.3.1 Distance measure selection

For determining the appropriate distance measures for simulation outcomes, I have considered several possible options for both the tail distance and the total comparison. To determine the best measure, I inspect how their quantifications correspond to the actual discrepancies seen from the simulation outcomes. Figure 4.8 shows the simulation outcomes of the β2AR system with various parameter sets (top two) and various initial concentration condition (bottom two). Solid line represent the results of simulation the original network. Dashed line shows the simulation outcomes of the network with 10% of error in the parameters, and dotted represents 60% of error in the parameters. The parameters under inaccuracy conditions are shown in Table 4.3. As for the initial conditions, Rs is the only component that has tokens initially, and therefore stimulates the system. This amount is increased by 10% (resulting in Rs having 11 tokens) and decreased by 60% (resulting with Rs having 4 tokens) to simulate the corresponding conditions. From the plots we observe that the inaccuracy

of 10% produces very little discrepancies with the original system and, therefore, we expect a quantification of this error to be small. Whereas 60% produces significant difference asking for larger quantification. We need to find the distance measure that correctly reflects this amount of introduced error.

Reaction	**Original**	**10% error**	**60% error**
$Rs \rightarrow R^*Ls$	500	520.5841	736.3780
$R^*Ls \rightarrow Rs$	4	3.8289	2.3818
$R^*Ls \rightarrow Rg^*Ls$	1.4	1.4036	1.8727
$Rg^*Ls \rightarrow R^*Ls$	0.036	0.0346	0.0488
$Rg^*Ls \rightarrow ArrRg^*Ls$	27	29.2238	28.7901
$ArrRg^*Ls \rightarrow Rg^*Ls$	4	4.0076	3.1256
$Rg^*Ls \rightarrow Rgs$	4	4.2166	2.6762
$Rgs \rightarrow Rg^*Ls$	500	472.2804	665.4132
$ArrRg^*Ls \rightarrow ArrRgs$	4	3.9679	3.9773
$ArrRgs \rightarrow ArrRg^*Ls$	500	502.9036	544.1009
$ArrRgs \rightarrow Rgs$	11	10.3863	7.4872
$Rgs \rightarrow Rs$	0.036	0.0328	0.0339
$ArrRg^*Ls \rightarrow ArrRgi$	0.22	0.21366	0.1034
$ArrRgi \rightarrow Rgi$	11	11.0562	5.4515
$Rgi \rightarrow Ri$	0.036	0.0375	0.0575
$Ri \rightarrow Rs$	0.09	0.0827	0.1037
$Rs \rightarrow Ri$	0.0085	0.0077	0.0128
$ArrRgs \rightarrow ArrRgi$	0.0085	0.0085	0.0039
$Rgs \rightarrow Rgi$	0.0085	0.0092	0.0118
$Rgi \rightarrow Rgs$	0.09	0.0884	0.0365
$ArrRgi \rightarrow Rdeg$	0.0001	0.00011	0.0001
$Rgi \rightarrow Rdeg$	0.0001	0.0001	0.00015
$Ri \rightarrow Rdeg$	0.0001	0.0001	0.00008

Table 4.3: Parameter sets for β2AR system with the parameter inaccuracies: The parameters that represent *one* instance of the system with 10% parameter inaccuracy, and *one* instance with 60% parameter inaccuracy. These instances are used to determine appropriate distance measures.

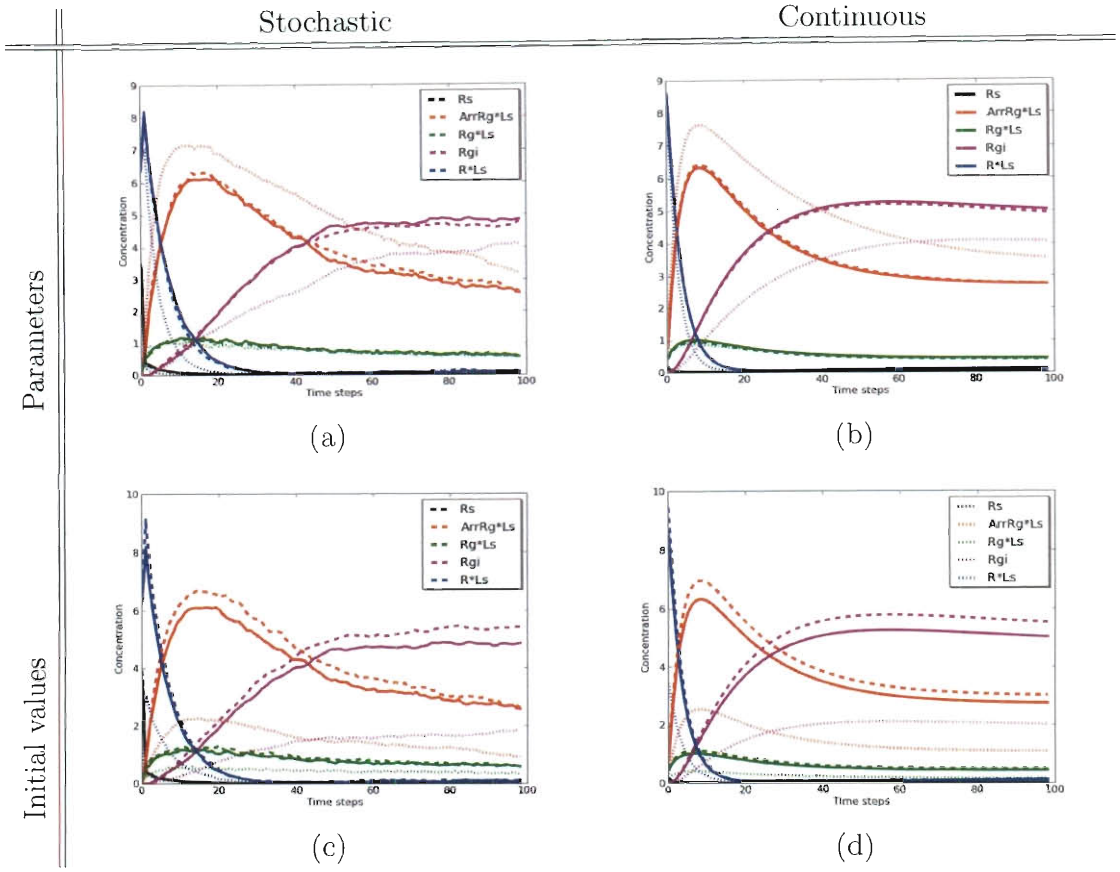


Figure 4.8: System comparison with varying noise: All images show comparison of β2AR system under different inaccuracy conditions. Images (a) and (b) correspond to parameter in accuracy. Images (c) and (d) correspond to initial concentrations inaccuracy. *Solid* line represent the system simulated with the original parameters. *Dashed* line shows the simulations results after 10% error was introduced into the system. *Dotted* line shows the simulation results after 60% error was introduced into the system. Image (a) and (c) show the stochastic outcomes, and (b) and (d) - continuous simulation results.

We are interested in two types of comparisons: tail and total distance between original trajectory and the trajectory generated by the system with error. I consider two measures for tail comparison: (1) tail point distance, and (2) batch means. In

the tail comparison we are given two time series Q and C both of length n. I find a point average of each tail and calculate the length of the tail segment of length k between them according to:

$$D(Q,C) = \left| \frac{\sum_{i=(n-k)}^{n} q_i}{k+1} - \frac{\sum_{i=(n-k)}^{n} c_i}{k+1} \right| \tag{4.1}$$

Finding the average of the tails, instead of considering just the last point of the simulation accounts for small fluctuations inherent in SPN simulations. Alternatively, batch means algorithm acquires the *interval* estimates of true steady state [52]. To do that the series is broken down into batches of size n. We, then, find the average of each batch and determine the confidence interval around the mean of these batches. While this measure considers the overall shape of the trajectory, if the simulation is long enough and reaches the steady state, it produces a tight bound around the true steady state behavior. Therefore, if the batch means interval is large, one can assume that at the time of measurements the protein might have not reached the steady state. In Figure 4.9 I show the error quantifications of each method for both deterministic (CPN) and stochastic (SPN) simulations on β2AR system. Top plot represents the quantification for parameter inaccuracy, and the bottom for initial values inaccuracy. The vertical black lines mark the instances of the impaired networks shown in Figure 4.8. For the network with 10% of parameter error we quantify 10%-15% of highest distance with *point tail* estimation and about 50% of the highest distance with *batch means*. As mentioned earlier, 10% of error produces very little deviation from the original network with *point tail* estimation providing more accurate quantification.

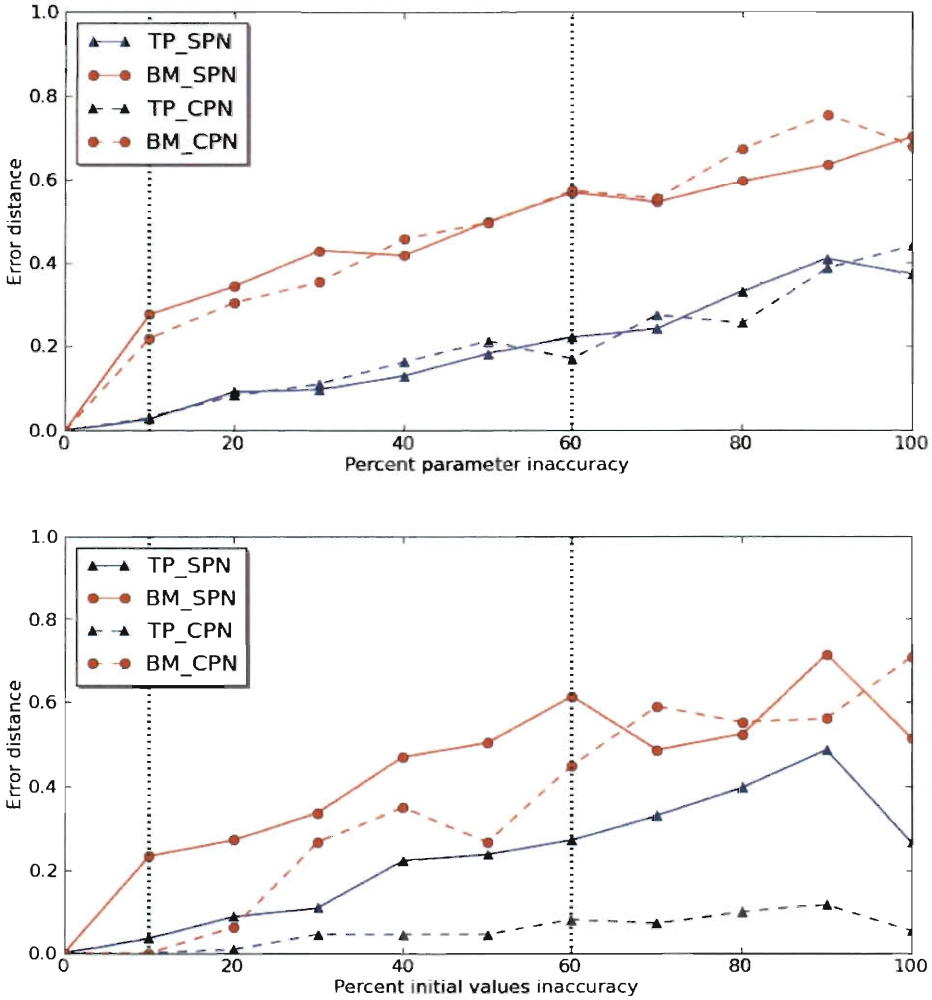
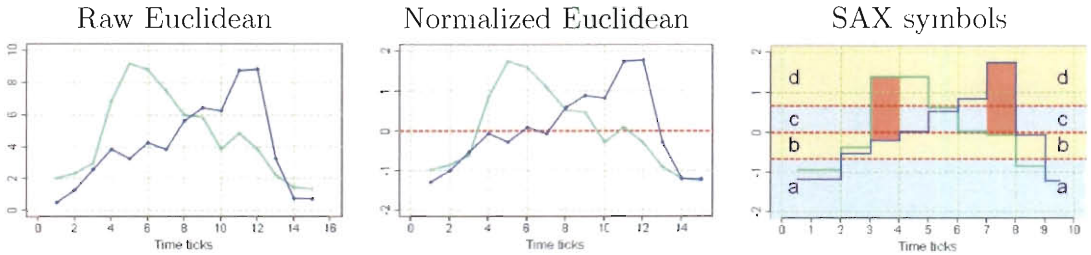


Figure 4.9: SPN vs CPN on steady state comparisons. *TP* represent the *tail point* approximation in both SPN and CPN. *BM* is the *batch mean* approximation. The curves result from averaging 10 error profiles. Top image shows how these measures perform with parameter inaccuracy. Bottom image shows error measure with initial values inaccuracy. Used β2AR system.



distance = 11.42126 distance = 4.170542 distance = 1.34

Figure 4.10: Illustration of different comparison measures: Left panel shows the raw points Euclidean distance comparison. Middle panel shows the time series after Z-normalization step and corresponding Euclidean distance measure. Right panel shows PAA segments of two time series with the assigned letter form 4 letter alphabet and the corresponding SAX score. Image taken from [53]

To measure the total distance between two trajectories I examine three different measures: (1) Euclidean distance on raw data, (2) Euclidean distance on Z-normalized data, and (3) SAX (Symbolic Aggregate Approximation) symbolic algorithm. In Figure 4.10 I show the example of all three measures with the resulting distances.

The most ordinary distance measure between two points can be defined the length of the line segment connecting them, and refers to as Euclidean distance. Comparing two time series Q and C of the same length n becomes calculating the distance between two points in Euclidean n-space, and defined by:

$$D(Q, C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2} \tag{4.2}$$

When two series possess similar features we expect the distance measure to be relatively small. However, if two series exhibit almost the same features but located far enough in space from each other, the raw Euclidean distance would produce rather

large quantification. To solve this we can apply Z-normalizations to each series. It transforms both vectors to have the mean of approximately 0 and the standard deviation (and variance) are in a range close to 1 [54]. This puts the series next to each other in space giving a more apples-to-apples comparison (middle panel of Figure 4.10). Nevertheless, the article by Lin *et al.* in [55] explains the reasons and solutions for some cases when the zero mean and unit standard deviation normalization fails. For example if a signal is constant over most of the time span with minor noise at short intervals, this normalization will overamplify the noise to the maximal amplitude. Also, if the time-series contains only single value and the standard deviation is not defined normalization will also fail.

Lin *et al.* propose SAX (Symbolic Aggregate Approximation) comparison measure and show that it is the lower bounding[2] approximation of the Euclidean distance which guarantees its correctness. The algorithms incorporates the normalization steps, followed by applying PAA method (section 3.4.1) to break it down into segments for symbol assignment from pre-defined alphabet of size a. The symbols are obtained by using Gaussian distribution properties. For a given alphabet of size a, this results in the set of "breakpoints" which is a sorted list of numbers $B = \beta_1, \ldots, \beta a - 1$ creating the equal areas under a Gaussian curve. The distance from β_i to β_{i+1} is equal to $1/a$ (β_0 and β_a are defined as $-\infty$and ∞, respectively). The time series is then converted into symbolic representation by assigning the letter from the region that PAA segement falls into (right panel in Figure 4.10). Using these breakpoints we also

[2]Lower bounding means the estimated distance in the reduced space is always less than or equal to the distance in the original space

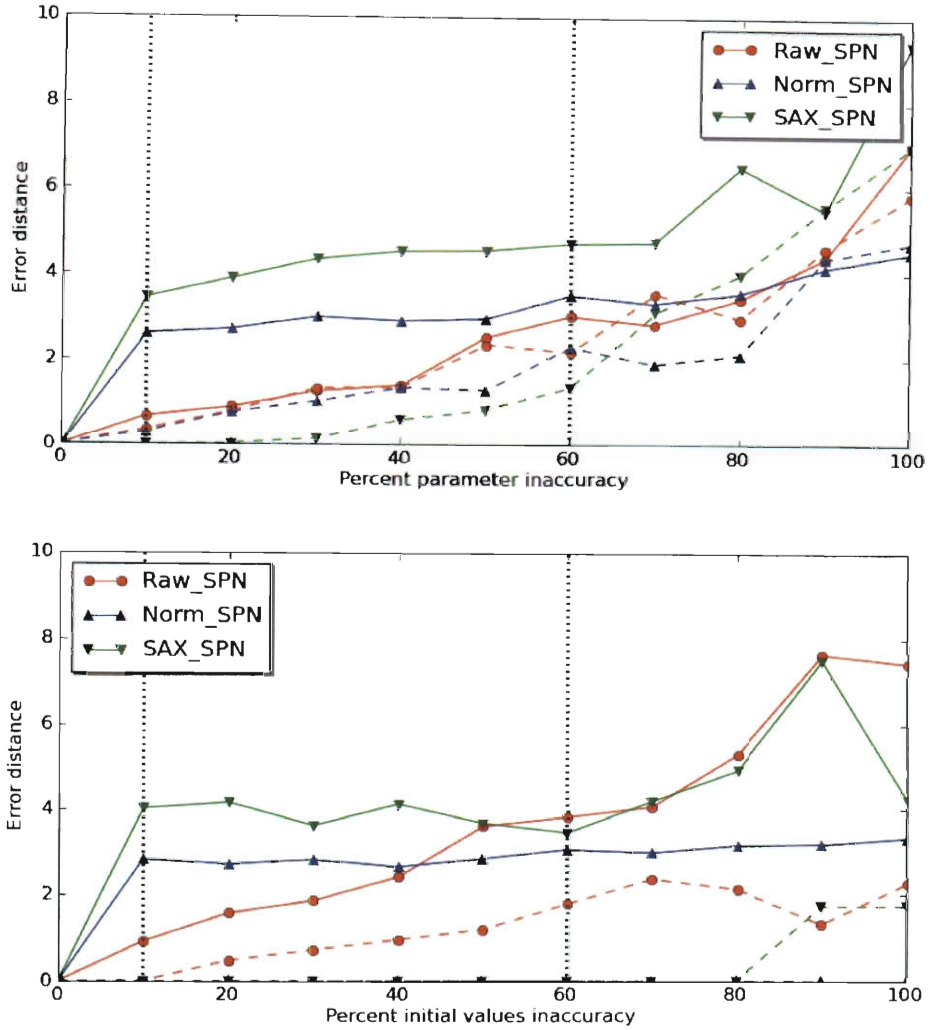


Figure 4.11: SPN vs CPN on transient comparisons. *Raw* represent Euclidean distance of raw time series. *Norm* refers to Euclidean distance comparison of normalized data series. *SAX* is the comparison with the symbolic algorithm. Solid lines represent the error when stochastic simulations are used. Dashed line denote the error of the deterministic simulaitons. The curves result from averaging 10 error profiles. Top image represent parameter inaccuracy. Bottom image represent initial value inaccuracy. Used β2AR system.

construct a lookup table for distance calculation. Below we show the example lookup table for an alphabet of size $a = 4$ (left) with the formula for calculating the values of each cell (right).

	a	**b**	**c**	**d**
a	0	0	0.67	1.34
b	0	0	0	0.67
c	0.67	0	0	0
d	1.34	0.67	0	0

$$cell_{q,c} = \begin{cases} 0, & \text{if } |q - c| \leq 1 \\ \beta_{max(q,c)} - \beta_{min(q,c)}, & \text{otherwise} \end{cases}$$

Once the table is acquired, we can calculate the distance between the series by adding up the distances from the lookup table.

In Figure 4.11, I show the performance of each algorithm for quantifying the error for both stochastic and deterministic simulators. Top plot is the results for the parameter inaccuracy and bottom for the initial values inaccuracies. Solid lines represent stochastic simulator and dashed lines correspond to deterministic outcomes. Line with circles denote results of raw time series comparison, triangles are normalized series comparison and upside-down triangles is SAX comparison. First thing to notice is that SAX provides zero error for deterministic simulator until about 80% of inaccuracy, followed by a spike to highest error quantification for both parameters and initial values. SAX behavior is more stable for stochastic simulator, but very different from that on deterministic. In SAX, its sporadic behavior and inconsistency in quantification make it not as promising measure for this work. Second thing to point out is the behavior of the normalized comparison, especially for the initial values inaccuracy. Introducing the error into the initial concentration causes shifts in

the trajectories. On the bottom plot, we do not see the curve for normalized comparison for deterministic simulations. This scenario corresponds to Figure 4.8(d), where normalizing these series would result in almost equivalent datasets with mean around zero (since only shifting in space occurred). This is exactly the case where normalized comparison fails as mentioned by Lin *et al.* in [55], making this measure not usable in this work.

The best quantification seems to be achieved by the raw Euclidean distance. In the case of 10% error, we expect low value of quantification, since its transient behavior is very close to the original (Figure 4.8). Raw Euclidean distance quantifies at about 10%-15%, from the highest amount of error whereas the other two comparison methods go as high as 70% of the highest or stay at zero. At 60% error, we see the expected amount of increase in the error for Euclidean distance as well. Based on these results, I choose raw Euclidean distance as the measure to quantify the dissimilarities in the total behavior. I summarized chosen distance measures below.

The tail distance equates the biologist taking a single measurement in the end of the experiment. To quantify the difference, we can find the distance between using Equation 4.1. Comparing more than just the end points but the averages of k end points allows to smooth out the small fluctuations inherent in the SPN simulations.

Total distance equates to the biologist making conclusions from the time series data. I calculate the total distance using Euclidean distance defined by Equation 4.2.

In Algorithm 4, I describe the entire error testing procedure. Steps 2 though 5 correspond to Figure 4.7.

Algorithm 4 Error Robustness Procedure

1. Execute original network and store it.
2. For each step s in range $(0, max]$:
   (a) Introduce error in the system in range $\pm[0, s]$
   (b) Simulate the model with inaccuracies
   (c) Using the original network, for each protein calculate the tail or total distance producing error vector $\boldsymbol{e_s}$
3. Create a single instance profile $\lambda = \{\boldsymbol{e_s}, \boldsymbol{e_{2s}}, \ldots, \boldsymbol{e_{max}}\}$
4. Go to step 2 and repeat m times.
5. Average m error profiles $\lambda_\mu = avg(\lambda_1, \lambda_2, \ldots, \lambda_m)$.
6. Find the average trajectory in λ_μ.

Chapter 5

Results

In the previous chapter I described the procedure for testing the robustness of Petri net simulators. I implemented a Java tool that executed the method, by testing the robustness with respect to (1) kinetic parameters, (2) initial concentrations, and (3) the topology. In this chapter, I analyze the outcomes of the testing.

5.1 Deliverable

At its core, the tool uses the *Petri Net Kernel* (PNK) for the models [56]. While PNK is a general infrastructure for building Petri net based tools, it allows for easy extensions. In fact, it already carries some necessary components to address biological networks, such as places, transitions, regular arc, enzyme arcs, and inhibition arcs. For the purposes of this projected I added several simulation extensions discussed in the following section.

5.1.1 Simulators

In the scope of this project I extended the general *FiringRule* class to support deterministic and stochastic execution. Deterministic approach, as described in Algorithm 1, is implemented within *Continuous* firing rule; the *Stochastic* firing rule implements the Gillespie algorithms described in algorithms 2 and 3. Both simulators use *rate* property on transitions to drive the dynamics of the system. In CPN it is used to calculate the rate of change for each species, and in SPN to determine propensities for calculating next reaction probability. These simulators will be tested for error robustness in this work. After introducing the error into the model, I am interested to execute it, and investigate the outcome of the simulations. Based on the results, I proceed to make observations about how the error values reflect the changes caused by the error.

5.1.2 The experimental setup

Original simulation: Both CPN and SPN were executed for 20 time units, which result in 20,000 time steps for CPN and variable number of time steps for SPN (also around 20,000). SPN instances are averaged over 200 runs. For more efficient comparison, each simulation is reduced to the size of 100 time steps using PAA algorithm (Section 3.4.1). The length of the simulation was chosen based on the visual inspection of the results. 20 time units was sufficient for all the time series to settle in the steady state in the original simulations.

Kinetic parameter testing: Kinetic parameters were varied within $[-x, +x]$

percentage range, where x is incremented by 10. The length of simulation was also increased proportionally to the testing range, or $20 + 20 \times x/100$. 200 single error profiles were created and averaged for the consensus profile.

Initial concentrations testing: Initial concentrations were varied within $[-x, +x]$ percentage range, where x is incremented by 10. The randomly chosen error values was rounded up to the integer since all the initial concentrations were required to be integers. If the initial concentration resulted in the value ≤ 0, it was set to 0. The length of simulation stayed the same (20 time units). 200 single error profiles were created and averaged for the consensus profile.

Topology testing: Topology testing was conducted in two ways - random removal of nodes (NRR) and random removal of edged (ERR). For both, I incrementally (by 1) removed x nodes/edges up to the maximum value of 6 nodes/edges. In NRR, I preserved the nodes that I built the error profiles for (never removed them). 200 single error profiles are created and averaged for the consensus profile. The error profiles included only the preserved species.

5.2 Results

In this section I analyze the results of the testing procedure described in Chapter 4. In all plots, we show tail distance results on the left panels and total distance on the right panels. For the tail comparison I use *tail point distance* (Equation 4.1), and for total comparison I utilize *raw Euclidean distance* (Equation 4.2). The

experiments are executed on two models: β2AR described in section 3.3 [46], and Erk/Mek pathway system described in section 4.1. For both, I calculate the average error distance of the whole system along with visualizing and analyzing the time series for the selected proteins. For β2AR, I analyze the surface receptors behavior: $Rs \leftrightarrow R^*Ls \leftrightarrow Rg^*Ls \leftrightarrow ArrRg^*Ls$ (species within the red square in Figure 3.3) along with the protein corresponding to the internalization of the treatment Rgi (circled species in Figure 3.3). In the Erk/Mek system, I continue analyzing the close dynamics of $AMPK$ guided by the phosphorylated form of AKT. I look at AKT, $AKT(u)$, $AMPK$, and $AMPK^*$.

In order to classify the behavior of individual proteins given the error distances, we define two measures. For each protein, we define Z_{tail} to denote the tail distance Z-score for an individual protein to the tail average in the consensus error profile; and Z_{total} to denote the total distance Z-score for an individual protein to the total average in the consensus error profile. The Z-scores is calculated for each error range according to the equations:

$$Z_{tail} = \frac{TD_i - TD_\mu}{TD_\sigma} \text{ and } Z_{total} = \frac{TOT_i - TOT_\mu}{TOT_\sigma} \tag{5.1}$$

where TD_i and TOT_i are the range specific raw distances for individual proteins, TD_μ, TD_σ and TOT_μ, TOT_μ are range specific mean and standard deviation for tail and total distances respectively. The example of calculating these scores is shown in Table 5.1. For individual protein, Z_{tail} and Z_{total} tells us how far its error is from the average. In the example in Table 5.1 $ArrRg^*Ls$'s error is consistently between 1.5

	Measure	20%	40%	60%	80%	100%
Steady	$ArrRg^*Ls$ raw TP	0.225	0.519	0.702	1.087	1.289
	Average raw TP	0.063	0.142	0.198	0.314	0.481
	St Dev of raw TP	0.085	0.19	0.27	0.43	0.53
	Z_{tail} measure	1.91	1.98	1.87	1.80	1.52
Trans	$ArrRg^*Ls$ raw Euc	2.795	4.734	7.321	10.015	16.000
	Average raw Euc	0.995	1.708	2.587	3.693	5.664
	St Dev of raw Euc	1.0	1.73	2.65	3.88	5.76
	Z_{total} measure	1.80	1.75	1.78	1.68	1.79

Table 5.1: The example of calculating measure Z_{tail} and Z_{total} For individual protein $ArrRg^*Ls$, the tail (raw tail point average, TP) and the total (raw Euclidean distance, Euc) distance measures are shown. They are calculated for systems with varying amount of *parameter* inaccuracies. Second line shows the average error for each error range. Third line is the standard deviation within each error range. Measures Z_{tail} and Z_{total} are calculated according to equations 5.1.

and 2.0 standard deviations from the average for tail comparisons and ~1.7 standard deviations from the total average distance across different error ranges.

5.2.1 Kinetic parameters

In β2AR model, Vayttaden *et al.* acquired a complete set of kinetic parameters through both the experimental work and the literature search. For the purpose of this work, we consider these parameter set to be accurate and complete. In [50] Iadevaia *et al.* computationally acquire a set of kinetic parameters for Erk/Mek system via *Particle Swarm Optimization* algorithm. I utilize these parameters, considering them accurate and complete for this study. In Figure 5.1 I show the error profiles for the parameter error testing. The x-axis corresponds to the percent of error introduced into the parameter set (percent deviation from the original parameter value). The

y-axis corresponds to the distance values for each protein in terms of tail and total distance.

In β2AR system, we see that for the both types of error proteins Rgi and $ArrRg^*Ls$ have the greatest distances. Rgi is the internalization of the treatment and $ArrRg^*Ls$ is the farthest surface protein. Since the system has only one initial concentration in Rs, by the time the signal gets to these proteins, the amount of error in the parameters accumulates enough that their distances are so large. Additionally, an interesting behavior can be observed in the error profile of R^*Ls: for the tail distance is really small and much below the average, whereas for the total distance it is now above the average (meaning $Z_{tail} \ll Z_{total}$). To validate the error values with the behavior of the system, I visually inspect the instances of the system with kinetic parameter error. In Figure 5.2, I show instances of 10% and 60% kinetic parameter errors for β2AR system. From the plots we see that the two proteins that have largest amount of error (Rgi and $ArrRg^*Ls$) show significant shift from its original behavior. Interestingly enough, the overall transient behavior is preserved. In the case of R^*Ls, where tail error is small and the total error is above average, we can explain it by the transient deviation (blue line in Figure 5.2), where R^*Ls shows faster drop, but settles in the original steady state.

If we approximate Z_{tail} and Z_{total} for $ArrRg^*Ls$, we get about the same value for both of them. If we approximate the Z-scores for Rgi, we get much lower value for Z_{tail} than Z_{total}. From these measures and observations I conclude that **(i)** if a protein shows $Z_{tail} \approx Z_{total}$, *its time series is most likely shifted, while keeping the same trend*; **(iii)** if a protein has $Z_{tail} \ll Z_{total}$, *its time series is most likely deviated*

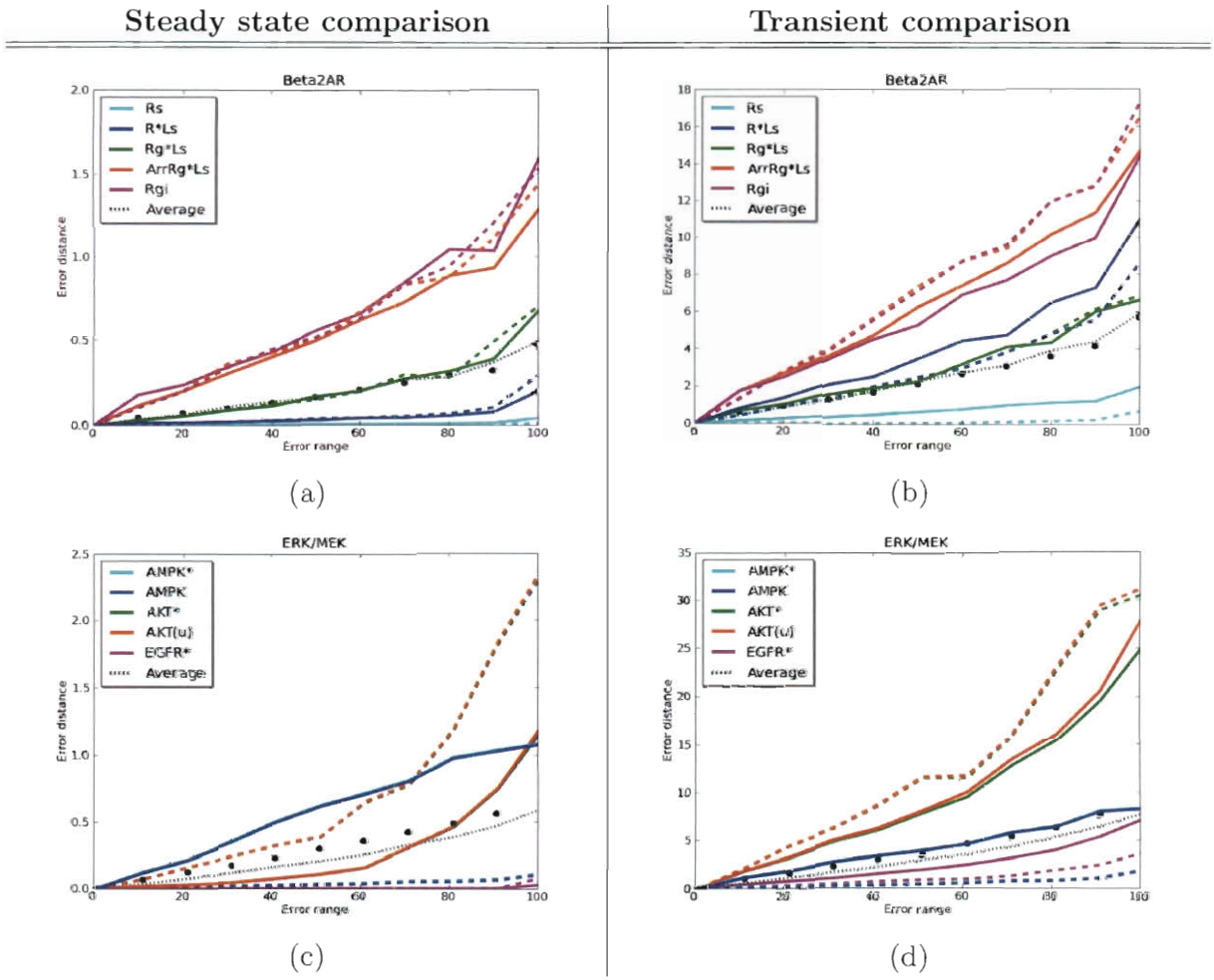


Figure 5.1: Parameter error robustness test. The plots show the error distances for selected proteins and their average. X-axis denotes the parameter deviation (percent of the original) allowed in the system. Y-axis is the distance measure resulted from simulating the system. Solid lines represent SPN (Gillespie direct method) simulation and dashed line are CPN (Continuous) simulation. Black circles (SPN) and black dotted line (CPN) represent the average of the distances for all species. The consensus profiles built from 200 single profiles. In the case of Erk/Mek model, we observe the overlap in error quantifications for the different forms of the same proteins. (AKT and $AMPK$).

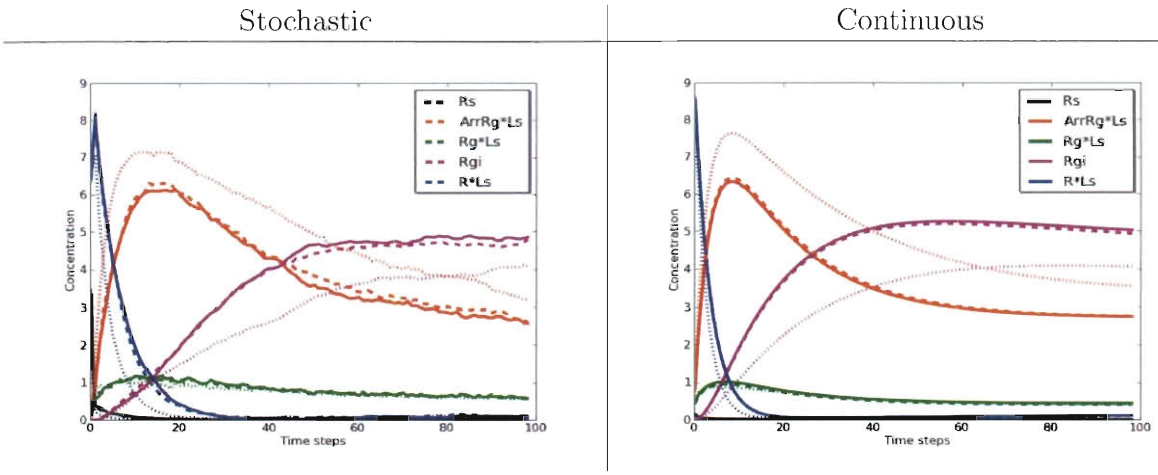


Figure 5.2: Instances of β2AR system with parameter error. *Solid* line represent the system simulated with the original parameters. *Dashed* line shows the simulations results with 10% parameter error. *Dotted* line shows the simulation results with 60% parameter error. We see shift proportional to the amount of error in the system.

in the transient behavior but eventually moves towards the original steady state.

In the results of Erk/Mek system, we see that different forms of the same protein completely overlap in their error values ($AMPK$ and $AMPK^*$; $AKT(u)$ and AKT^*). This results is intuitive, since the mass is preserved between those forms and the error is expressed to the same extend in both. Additionally, we observe the $AMPK$ errors being significantly higher for SPN than for CPN. Recall, that Figure 4.2 showed that the behavior of $AMPK$ is affected by the zero-concentration of AKT^*, with SPN and CPN producing different results. In Figure 5.3, I compare these proteins between *one* instance of the 100% parameter error networks and the original network. Solid lines represent the original concentrations and dashed lines are concentrations from the inaccurate system. These plots show that with the new kinetic parameter

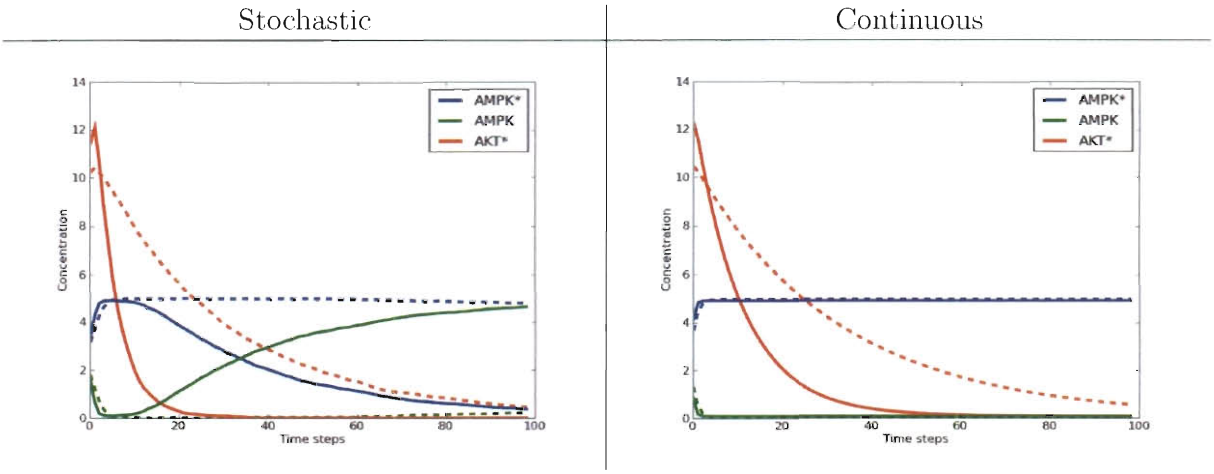


Figure 5.3: AMPK behavior in the system with parameter error. *Solid* line represent the system simulated with the original parameters. *Dashed* line shows the simulations results from simulating *one* instance of 100% parameter error system. The system with inaccurate parameter now resembles CPN original simulation due to the fact that AKT^* does not deplete its concentration in the allocated amount of time.

set, AKT^* does not reach the zero-concentration in the allocated amount of time. This is reflected in the behavior of $AMPK$. In CPN where $AMPK$ is not affected by AKT^* originally (and, therefore overlaps now), it produces very small amount of error. In SPN, where originally zero-concentration of AKT^* de-activated $AMPK$, the system with error completely misinterpreted the behavior at time of measurements (tail distance). From this behavior we can make last observation: **(ii)** if a protein has $Z_{tail} \gg Z_{total}$, *its time series is most likely neither reach the accurate steady state nor captured the overall trend in the allocated amount of simulation time.*

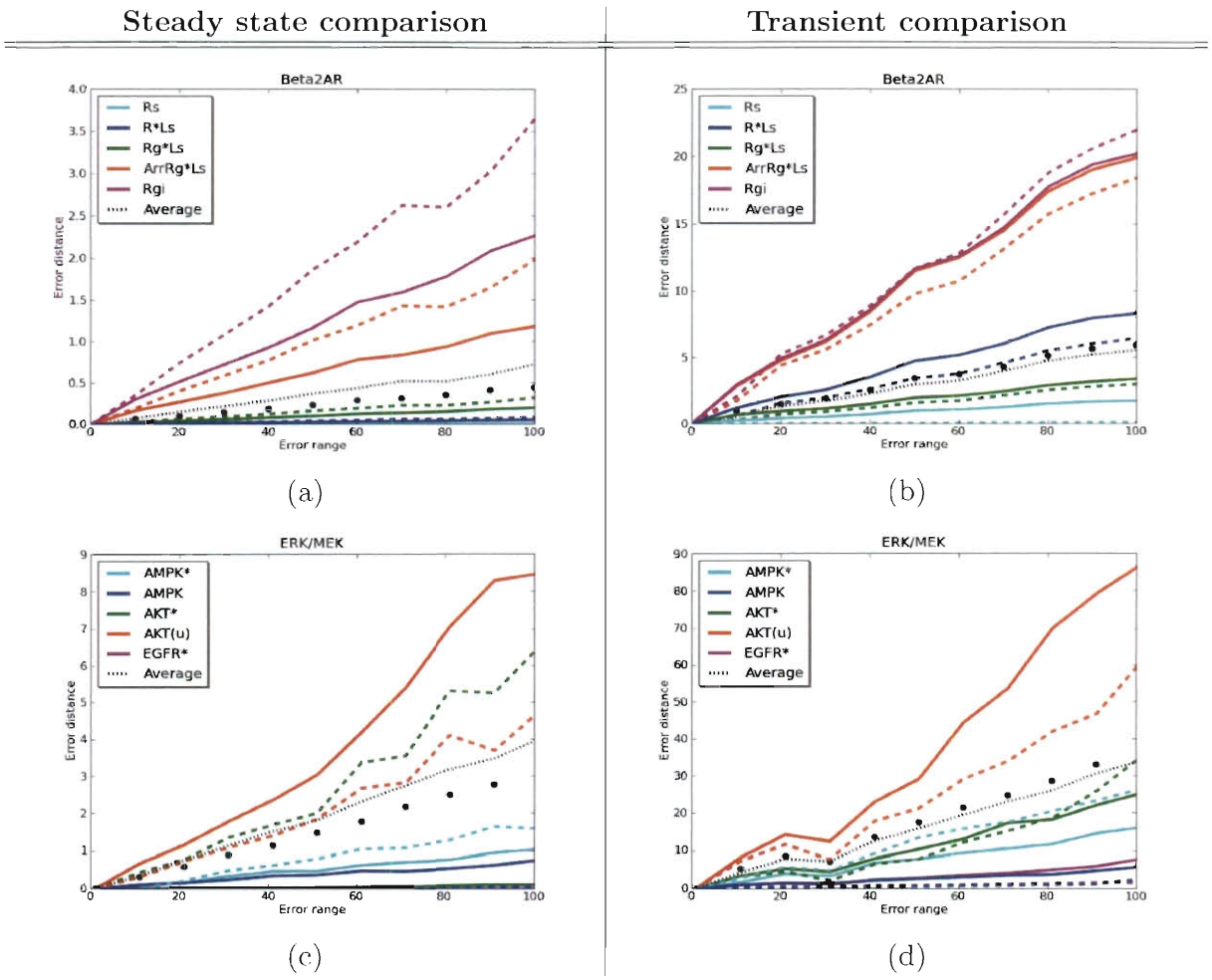


Figure 5.4: Initial values error robustness test. The plots show the error distances for selected proteins and their average. X-axis denotes the initial values deviation (percent of the original) allowed in the system. Y-axis is the distance measure resulted from simulating the system. Solid lines represent SPN (Gillespie direct method) simulation and dashed line are CPN (Continuous) simulation. Black circles (SPN) and black dotted line (CPN) represent the average of the distances for all species. The consensus profiles built from 200 single profiles.

5.2.2 Initial values

In this study, the initial concentrations are assigned from the experimental data. Generally, in experimental data, concentrations are represented by small (decimal) amounts. I show that taking initial concentrations from the experimental data, multiplying all of them by the order of magnitude to reach integer amount of tokens is sufficient to produce accurate simulation results. In the context of the initial values study, the two systems under considerations differ significantly. In β2AR, there is one single input with only protein Rs having non-zero concentration. In Erk/Mek system almost all proteins are present in some form, with greater amount in the receptor proteins. In Figure 5.4 I show the results of error testing on initial conditions.

For β2AR system we see the similar error profiles, where Rgi and $ArrRg^*Ls$ have the greatest amount of error. Overall, we observe higher error values, which is reasonable, since by changing the initial concentration of the receptor protein, we modify the total amount of the mass in the system and, therefore, shifting the time series more significantly. Examining instances of the systems with initial concentrations error in Figure 5.5, we see that changing initial concentrations in the system still preserves the trends very well, but can have very significant impact to the error results since the overall mass in the system changes.

In Erk/Mek system, the behavior of $AKT(u)$ stands out with exceptionally high error profile. This could be due to the fact that during the testing, we introduce mass into initially empty $AKT(u)$. Since the rates towards this inactive state is high, this mass never escapes differentiating it from the original behavior for the length of the whole simulation. Additionally, the AKT^* has very high (above the average)

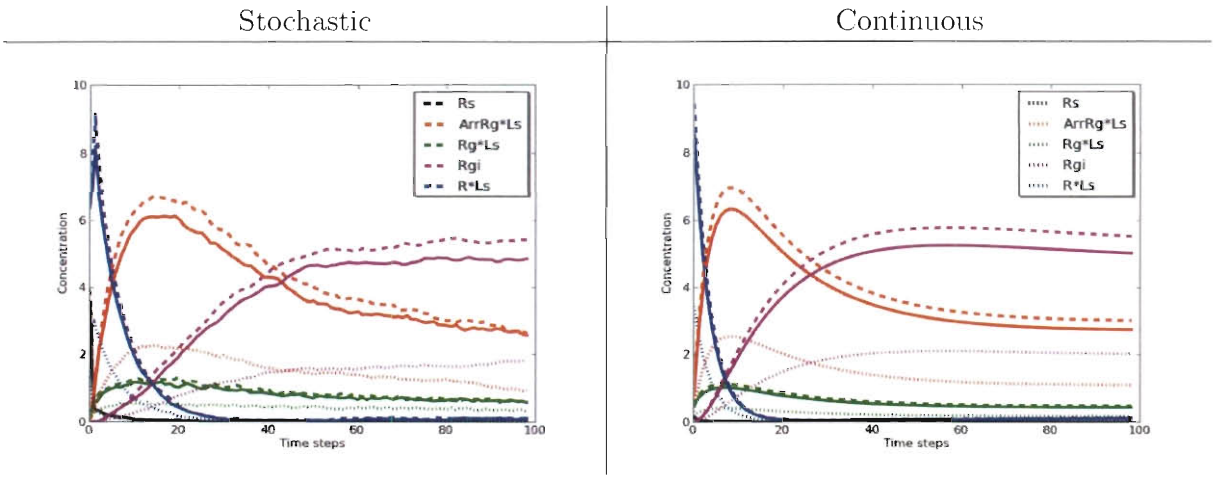


Figure 5.5: Instances of β2AR system with parameter error: *Solid* line represent the system simulated with the original parameters. *Dashed* line shows the simulations results with 10% parameter error. *Dotted* line shows the simulation results with 60% parameter error.

tail error and relatively small total error for SPN. This would be classified it as (ii). This could be due to the same reason as we have seen with parameter testing. The sources of APK phosphorylation ($EGFR^*$ and $IRS1^*$) do not deplete in the allocated amount of time, therefore, it does not start loosing its concentration as in the original simulation.

5.2.3 Topology

In the topology testing I am interested to see how the missing interactions affect the behavior of the system. Particularly, I test missing nodes (species) and transitions (reactions). I remove up to 6 nodes/edges. This number of missing interactions is rather significant for β2AR system (with total 10 species and 23 reactions). But for

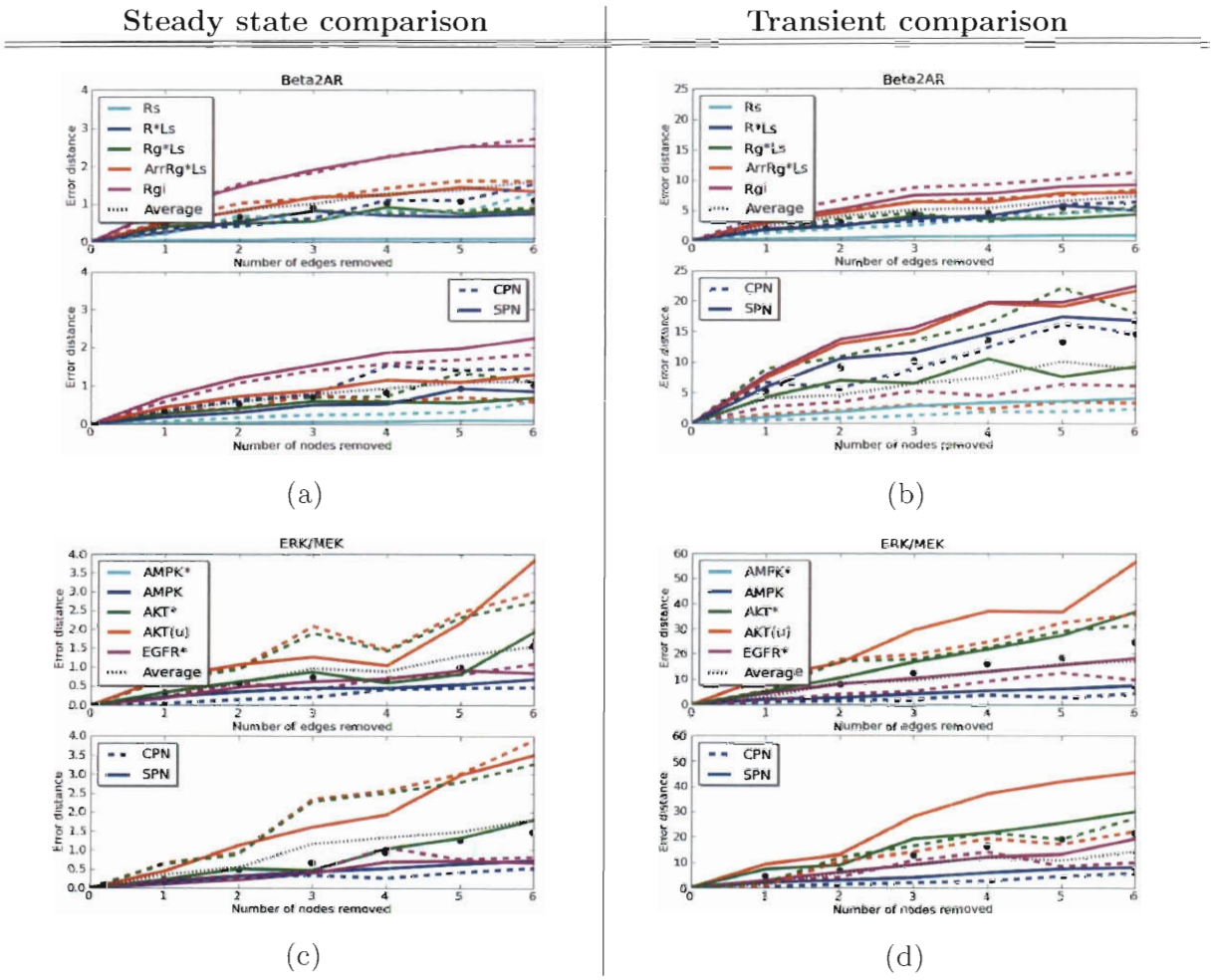


Figure 5.6: Topology error robustness test: The plots show the error distances for selected proteins and their average. X-axis denotes the topology inaccuracy (number of randomly removed edges/nodes) allowed in the system. Y-axis is the distance measure resulted from simulating the system. Solid lines represent SPN (Gillespie direct method) simulation and dashed line are CPN (Continuous) simulation. Black circles (SPN) and black dotted line (CPN) represent the average of the distances for all species. The consensus profiles built from 200 single profiles.

Erk/Mek system, I expect 6 missing nodes/transitions make less of an impact on the simulations (with total 40 species and 47 reactions). Random node removal is restricted to ignore the species under investigation (the ones being plotted), in order to be able to calculate the error at every run. In Figure 5.6 I show the results of topology testing for both tail and total distances.

In β2AR system, right away we see the error values start linearly increase with the amount of error in both nodes and edges. We see that for random edge removal Rgi (internalization) exhibits trend (ii) – $Z_{tail} \gg Z_{total}$. Since there are no direct interaction between the surface and Rgi, this is the expected behavior since by removing the edges we are cutting off pathways for the treatment internalization.

In Erk/Mek system we do not see smooth linear increase in the error distances with the number of missing nodes/edges. In fact, we see spikes. For example, in the random edge removal for up to 5 edges, we see relatively flat error profile, but see a spike when we go to 6. Similarly for the nodes, we see a spike when going from 2 to 3 and from 5 to 6. This network is much larger and is robust to the small number of missing interactions, as it can make up for it with alternative pathways. Additionally, we see that almost all the species have approximately the same Z-scores, meaning that their time series are shifted without significant deviations in steady state or transient behavior. This behavior seems reasonable since removal of 6 entities does not effect the system of this size nearly as much. Additionally, the displayed species are clustered and if we remove a node that is far away from this cluster, it might not have significant effect on the error of the observed proteins. Therefore, all of them would be expected to exhibit trends (i) or (iii) – shifting in steady state or overall

trend.

5.3 Observation validations

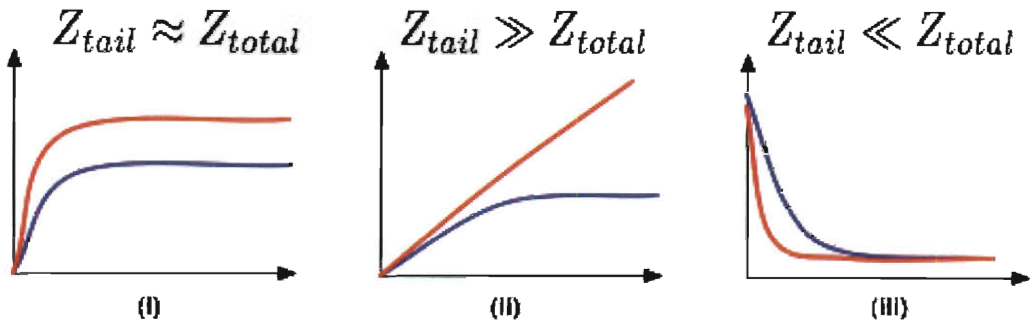


Figure 5.7: Illustration of the observations made in Chapter 5. (i) trend shift; (ii) neither trend was preserved nor was able to reach the steady state in the allocated amount of simulation time; (iii) transient trend changes, but steady state is preserved. Categories (ii) and (iii) arise from the difference of some threshold value.

While examining Petri net robustness to the inaccuracies in the model, I made three observations. Given the measures Z_{tail} and Z_{total}, these observations predict how the error in the model affects the time series. In Figure 5.7 I show the illustration of the expected trends given the relationship between Z_{tail} and Z_{total}. To validate the classification categories, I examine the classification calculated from the consensus profiles with respect to *one* instance of the 100% kinetic parameters inaccuracies system (100-PI system). Using just the last points in the error profiles (representing 100% error), I calculate the Z-values and use them to assign the classification to each proteins. I assign categories (ii) and (iii) when at least one simulation strategy shows difference larger than some threshold, in this case ≥ 0.4 standard deviations. This

Name	Class	CPN		SPN	
	≥ 0.4	Z_{tail}	Z_{total}	Z_{tail}	Z_{total}
Rs	(i)	-0.81	-0.79	-0.79	-0.66
ArrRg*Ls	(i)	1.57	1.62	1.41	1.57
Rg*Ls	(i)	0.35	0.14	0.33	0.15
Rgi	(ii)	1.72	1.74	1.95	1.53
R*Ls	(iii)	-0.34	0.42	-0.5	0.92
Average		**0.49**	**5.86**	**0.48**	**5.73**

Table 5.2: Kinetic parameters Z_{tail} and Z_{total} classifications for selected proteins in β2AR system. Classification is done using the threshold of ≥ 0.4.

threshold is established by visually analyzing the time series data.

In β2AR system, I continue analyzing the behavior of the surface proteins and the response of the internalization protein. In Table 5.2, I show the Z-scores for 100% error case and their classification for selected proteins. Only *Rgi* (internalization) exhibits behavior (ii), meaning $Z_{tail} \gg Z_{total}$. This classification signifies that this protein potentially did not reach its steady state in the allowed amount of simulation time (while I ensure that the original simulation *does* reach the steady state in the allocated time interval). For the classification based on the initial concentration error can be found in Table A.1. The tables A.2 and A.3 show the topology incompleteness classifications, where only *Rgi* is (ii) since with missing topology, we break pathways towards internalization.

In the case of Erk/Mek model, I analyze the sub-topology associated with the activation of $AMPK$. In Figure 5.8, I show the explicit Petri net representation of its dynamics and the simplified reactions associated with this sub-network. In Table 5.3, I show the classification resulted from 100% error distances for the selected proteins.

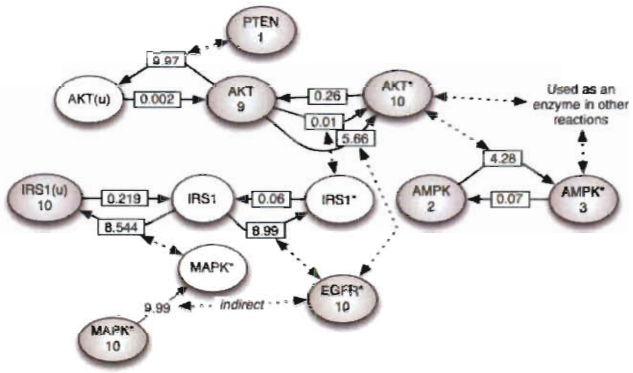
Explicit representation	Simplified reactions
	$EGFR \rightsquigarrow MAPK$ $EGFR \rightarrow AKT$ $EGFR \rightarrow IRS1$ $MAPK -\mid IRS1$ $IRS1 \rightarrow AKT$ $PTEN -\mid AKT$ $AKT \rightarrow AMPK$

Figure 5.8: Erk/Mek model sub-topology for observation validation: The sub-topology of Erk/Mek model that leads to activation of $AMPK$. The left panel shows the explicit Petri net model with all the kinetic parameters. The left panel shows simplified reactions associated with the network. $\rightsquigarrow$ represents indirect activation. $\rightarrow$ represents direct (enzymic) activation. $-|$ represents enzymic inhibition.

As we have seen with $AMPK$, it classifies as (ii) with the stochastic simulator because AKT^* ($AMPK$'s activator) does not get fully exhausted in the allocated amount of simulation time (as was shown in Figure 5.3) and, therefore, $AMPK$ never starts going towards its *original* steady state. Similarly, the classifications due to the initial concentrations values can be found in Table A.4. In tables A.5 and A.6 we see the classification resulted from the random node and edges removal respectively.

The peculiarity of this testing procedure is that while I introduce the error in the system, I vary each parameter up to 100% of its original value, but keep the overall relativeness the same. For example the parameter $p_1 = 500$, can vary in range $[0, 1000]$ whereas the parameter of $p_2 = 0.001$ would be within $[0, 0.002]$, with p_1 being consistently greater than p_2. These two parameters lie on the opposite side of the

Name	Class	CPN		SPN	
	≥ 0.4	Z_{tail}	Z_{total}	Z_{tail}	Z_{total}
AKT(u)	(iii)	2.19	2.58	0.54	2.03
AKT	(iii)	-0.67	-0.69	-0.66	-0.22
AKT*	(iii)	2.13	2.51	0.5	1.7
IRS1(u)	(i)	1.61	1.47	-0.23	-0.26
IRS1	(i)	-0.67	-0.8	-0.42	-0.6
IRS1*	(i)	1.61	1.49	-0.48	-0.62
AMPK	(ii)	-0.6	-0.66	0.39	-0.13
AMPK*	(ii)	-0.6	-0.66	0.39	-0.13
MAPK	(iii)	-0.73	-0.44	-0.68	-0.16
MAPK*	(iii)	-0.73	-0.44	-0.68	-0.16
EGFR*	(iii)	-0.63	-0.45	-0.67	-0.24
PTEN	(i)	-0.73	-0.87	-0.7	-1.02
Average		**0.59**	**7.84**	**0.69**	**9.47**

Table 5.3: Kinetic parameters Z_{tail} and Z_{total} classifications for selected proteins in Erk/Mek system. Classification is done using the threshold of ≥ 0.4.

spectrum and do not overlap. This means that all parameters that execute at original rate < 0.5 will be less than 1 even up to 99% error; and the parameters with original rates of > 1 will stay greater then 1 up to 99% of inaccuracies. The networks used in this study, I separate the parameter values at the boundary of 1, dividing them into low and high. Using this division, *can we create a binary abstracted model with fast and slow reactions and produce correct qualitative results?*

5.4 Parameter abstraction

The issue of the reaction rate parameter approximation is an eminent one in the world of signaling network modeling. Generally rate parameters are difficult to

acquire. They can be experimentally or computationally approximated, or researched in the literature [51, 50]. The examples of non-parametric approaches has been found in the literature. In [57], Albert *et al.* suggest a non-parametric for gene control networks. In [58] Ruths *et al.* suggests a non-parametric approach for qualitative modeling of signaling networks. While the kinetic parameters were not needed, the initial concentrations still had to be approximated experimentally. In this section, I am investigating how different simulators respond to the parameter abstraction in the model. Using Erk/Mek system I create fully binary model and investigate how it differs from the fully parameterized model in terms of protein classifications for error. To make the system binary, I have to identify two values to represent *high* and *low* kinetic parameter rates. Every kinetic parameter in the system falls in one of the binary categories: $Lo \in (0, 1]$ and $Hi \in (1, \infty)$. The new binary parameters is now the averages of each set[1] $\mu(Lo) = 0.06$ and $\mu(Hi) = 7.62$. To assigned the initial values, I give 10 tokens to the species that are present in the system and 0 tokens to those of 0 concentration in the beginning of the simulation. It is true that the final concentrations of species will not be of non-binary nature, which, in turn, would aid in better qualitative analysis.

In Table 5.4 I show the Z_{tail} and Z_{total} values for the error distances resulted from comparing original network to the binary abstraction model. It is interesting to notice that only two species classified as (ii) and no species classified as (iii). In addition, $AMPK$ did not classify as (ii) by SPN as it did with the regular error testing (in 100-PI). In Figure 5.9 I compare an instance of a model with 100% inaccuracies in

[1] mathematical set

Name	Class	CPN		SPN	
	≥ 0.25	Z_{tail}	Z_{total}	Z_{tail}	Z_{total}
AKT*	(ii)	1.37	0.91	-0.31	-0.31
AKT	(i)	-0.68	-0.68	-0.69	-0.59
AKT(u)	(ii)	1.17	0.71	-0.52	-0.24
AMPK	(i)	-0.68	-0.71	-0.60	-0.59
AMPK*	(i)	2.55	2.62	3.01	2.80
IRS1(u)	(i)	-0.19	-0.40	-0.56	-0.65
IRS1*	(i)	-0.20	-0.40	-0.52	-0.61
MAPK	(i)	-0.70	-0.65	-0.73	-0.54
MAPK*	(i)	-0.70	-0.65	-0.73	-0.54
IRS1	(i)	-0.69	-0.73	-0.69	-0.75
EGFR*	(i)	-0.66	-0.67	-0.69	-0.60
Average		**3.090**	**31.364**	**3.094**	**30.561**

Table 5.4: Z_{tail} and Z_{total} values for individual proteins in MEK/ERK binary model. Classification is done using the threshold of ≥ 0.25.

kinetic parameters (100-PI) and the binary abstracted model (BA).

I plot the forms of AKT (those classified as (ii)) along with the proteins in its immediate surrounding. $IRS1^*$ activates AKT and $AMPK^*$ is being activated by AKT. From the plots we can see that CPN classified forms of AKT as (ii). The rates that activate and de-activate AKT are the same in the BA model, therefore, CPN can not represent the crossover of concentration towards greater un-phosphorylation. SPN, however, is capable to capture this behavior. While SPN BA models seems to perform relatively well for the selected proteins, there are very large shifts that happen due to the abstraction of the initial concentrations and more investigation is needed to conclude on its qualitative correctness. For example, the concentration os $AMPK^*$ is shifted to have the highest concentration, whereas in the original simulation is was

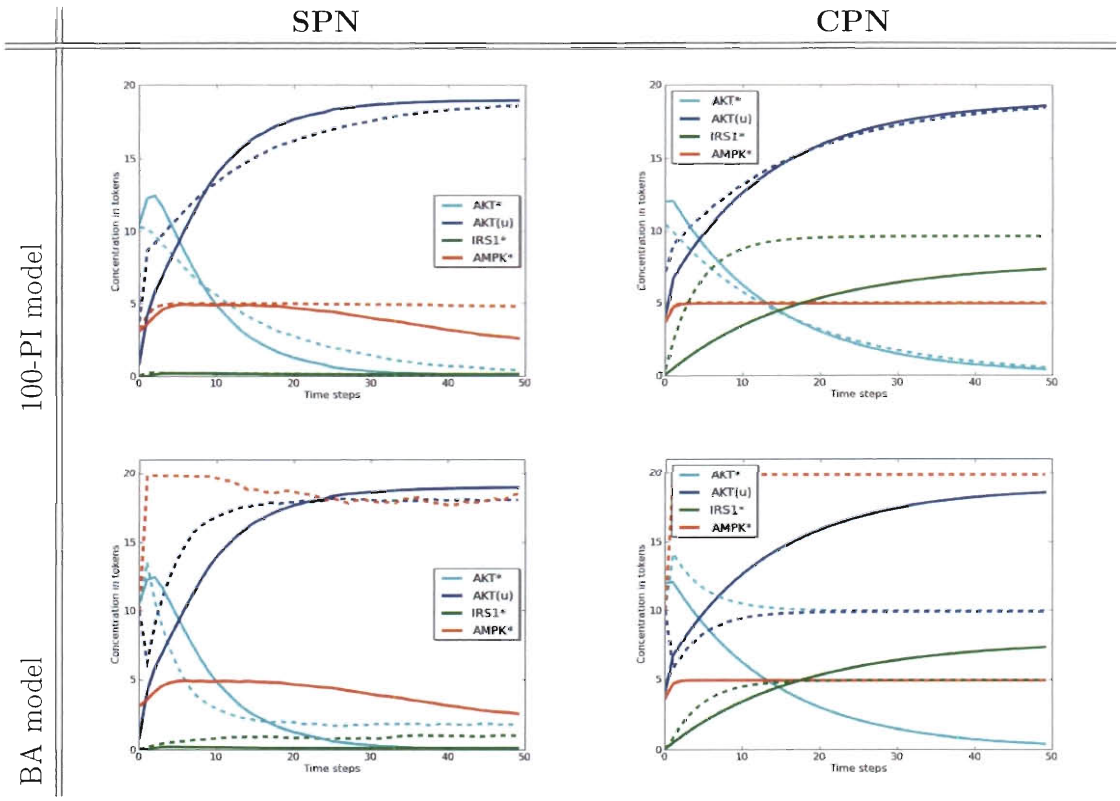


Figure 5.9: Observation classification validations from 100-PI and BA models: Top panels show the validations of classification for 100-PI (100%-parameter inaccuracy) model. Bottom panels show the validations of classification for BA (binary abstraction) model. Solid lines original simulations. Dashed lines are simulations of perturbed system. SPN BA model corresponds sufficintly well to the original data.

staying relatively low comparing to other species.

In this chapter, I showed the results of the error robustness testing procedure. Based on the results, I made the observations that allow to classify error effects on simulation results in three categories: (i) same trend, shift in the steady state ($Z_{tail} \approx Z_{total}$). (ii) loss of trend and does not reach the steady state in the given amount of

time ($Z_{tail} \gg Z_{total}$), and (iii) shift in trend, same steady state ($Z_{tail} \gg Z_{total}$). Classification to categories (ii) and (iii) arise when the difference between Z_{tail} and Z_{total} is ≥ 0.4 standard deviations. I generated classification from the consensus error profiles and validated that the classifications actually correspond to the expected behavior by examining a *single* instance of 100-PI system. Similar classification was done with the initial concentrations inaccuracies and topology incompletions. The corresponding tables can be found in the supplementary material. This measure can help biologist to identify the species that under perturbation did not reach the steady state (ii), or significantly shifted in their concentrations. Additionally, I created a binary abstracted model to test the importance of the parameters in the signaling network. While binary model preserves most of the trends of the system, the shifts in the trajectories appear to be rather large resulting in the invalid qualitative results.

Chapter 6

Conclusions

In this work, I investigated the use of Petri nets for modeling signaling networks and their robustness to the inaccuracies commonly present in the signaling pathway models. The contribution of this thesis are depicted in Figure 1.5. **First**, I introduced the Petri net framework and showed how to convert a signaling network into a Petri net and parameterize it appropriately. Particularly, I showed that taking the kinetic parameters and the initial concentrations from experimental data produced sufficiently good simulation results (Chapter 3). **Second**, I described a procedure for systematically introducing error into the system and proposed ways to measure the effects of this error on the simulation results (Chapter 4). I test the signaling network robustness to the error in three categories (a) kinetic parameters, (b) initial values and (c) topology incompleteness (nodes and edges). After simulating the system with error, for each protein I compare its time series with the original time series. I use two metrics for comparison: (1) tail point distance and (2) total time series distance. The

first one corresponds the biologist taking a single measurement after the experiment, while the second one corresponds to analyzing the time series data. Based on these distances, I defined two measures that can be calculated for each species: Z_{tail} - the number of standard deviations away from the averaged tail error in the consensus profile; and Z_{total} - the number of standard deviations away from the averaged total error in the consensus profile. From the relationship between Z_{tail} and Z_{total} I can predict how the error effects the behavior of the individual proteins.

The classification falls into three categories: (i) same trend, but shifting in the steady state ($Z_{tail} \approx Z_{total}$). (ii) loss of trend and potentially not reaching the steady state in the allocated time ($Z_{tail} \gg Z_{total}$), and (iii) shift in trend, but eventually settling in the steady state ($Z_{tail} \gg Z_{total}$). The least favorable classification is (ii). The inaccuracies in the parameters seem to modify the behavior of the system to the greatest extend, where as the inaccuracies in the initially concentrations mainly shift the trajectories proportionally to overall mass change in the system.

Third, I created a binary abstracted model to investigate how much information can be captured by the Petri net from the network topology only (Section 5.4). The results have shown that binary abstracted (BA) model produces similar trends but creates significant shifts in the steady state concentrations that potentially can lead to incorrect qualitative conclusions.

6.1 Future work

The overarching goal of this work is to demonstrate that Petri net is a good framework for the integrated model of the whole-cell function. As I showed in Section 1.1 Petri nets have great potential for representing whole-cell behavior. In the context of metabolic networks, there is a straightforward mapping from the stoichiometry associated with the reactions to the Petri net structure. Various static analysis methods for qualitative Petri nets (described in Section 2.2.1) has been extensively applied to the models of metabolism. For regulatory networks, I showed the conversion for the Boolean methodology, which is widely accepted and used method for modeling regulation, to the Petri nets. Petri net representation of regulation overcomes some of the shortcomings of the Boolean networks: each gene can now be represented by more than two values, reflecting the strength the expression, not just the on/off nature; the transition in the network can now execute asynchronously, providing a better interface for integration. For signaling, the common way of modeling is to construct a set of ODEs and simulate its dynamics by solving them over time. In this work, I show how continuous Petri nets (CPNs) is just a graphical representation of ODEs. In addition PN framework allows for stochastic simulations of signaling networks via SPN. In this work, I show that PN is a good framework for modeling signaling networks. Particularly, in conjunction with experimental data, we can devise a model that accurately captures the dynamic of the signaling system.

To this day, there has been some efforts on integration in the literature. Lee *et al.* proposed idFBA method that combines metabolic, regulatory and signaling networks

within FBA-based framework [59]. In this approach they construct a stoichiometric matrix, and optimize it for a set of constraints that are based on regulation and signaling. After the optimization, they modify the constraints to proceed with the optimization again. In this method, the system goes though the set of states towards the optimal solution. This, however, does not truly portray the fully concurrent dynamics of the system, just solves it for the steady state as the constraints change. Similarly, Shlomi *et al.* studies the extent to which the regulatory constraints match the flux activity states in the SR-FBA method. The author applies a mixed integer linear programing (MILP) to find a steady state flux solution using regulation activity as a constraint [60]. Covert *et al.* propose iFBA integration method similar to the idFBA, but he specifies the constraints that come out of regulation and signaling by solving their native methods [61]. They use Boolean network to calculate regulatory activity (gene and protein expression) and ODEs to solve for signaling constraints. These are used in the FBA linear programming problem. All these methods solve for the steady state of the system by optimizing FBA constraint matrix using regulation and signaling as constraints. With Petri nets we can represent concurrent dynamics of the system.

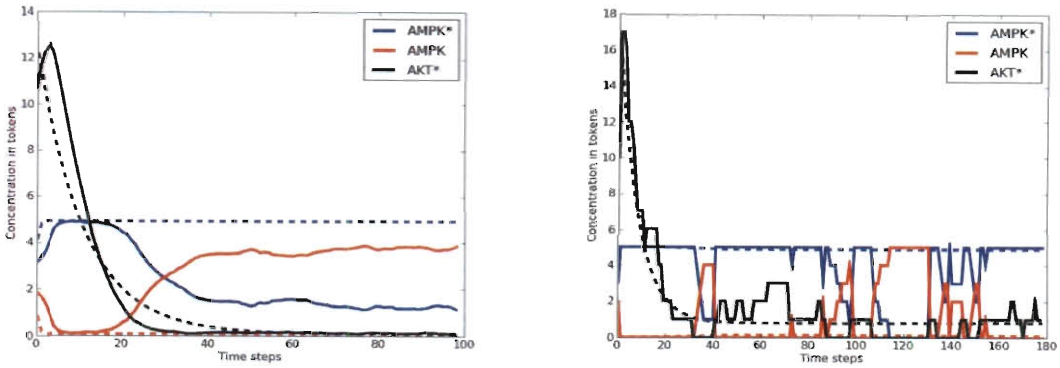
In conclusion, using Petri nets to create the integrated model is a promising approach. Petri nets have shown to be successful in the modeling of metabolic and regulatory networks. This thesis fills in the gap by demonstrating that this approach is also suitable for signaling pathways.

Appendix A

Supplementary material

Figure A.1 shows the sensitivity of deterministic simulators to zero-concentrations (described in Section 4.1). The immediate dynamics of $AMPK$ was simulated in isolation. Left panel of Figure A.1 shows the Petri net simulators: CPN for deterministic and SPN for stochastic. As seen from the graph, when the concentration of AKT^* is exhausted SPN captures it by moving mass of $AMPK$ from phosphorylated state to de-activated state. Whereas in the CPN, changes happen at such negligible amounts that the system seems to stay at the initial state. On the right panel similar behavior can be seem from the simulators available in COPASI, where LSODA used to simulate deterministically and Gillespie direct method for stochastic. Deterministic simulator stays in the original states, whereas *single instance* of Gillespie algorithm shows the the change in steady states (which would become more obvious with averaging).

Table A.1 shows the classification for the β2AR system for the 100% initial concentration inaccuracies consensus profile. Internalization protein Rgi classifies as (ii).



Petri net simulators COPASI simulation

Figure A.1: Sensitivity of zero-concentraitons of deterministic simulators: *Solid line* represent stochastic simulators. *Dashed lines* represent deterministic simulators. In the studied topology AKT^* is the only activator of $AMPK$, and, therefore, when the concentration of AKT^* depletes, we expect $AMPK$ to start de-phosphorylating. This behavior is not captured by deterministic simulators (dashed lines in both PN and COPASI), where $AMPK^*$ stays high. However, in SPN we see $AMPK$ moving towards deactivated state.

Name	Class	CPN		SPN	
	≥ 0.4	Z_{tail}	Z_{total}	Z_{tail}	Z_{total}
Rs	(i)	-0.59	-0.67	-0.59	-0.53
ArrRg*Ls	(iii)	1.03	1.59	0.98	1.78
Rg*Ls	(i)	-0.33	-0.32	-0.33	-0.32
Rgi	(ii)	2.4	2.03	2.43	1.81
R*Ls	(iii)	-0.53	0.11	-0.52	0.31
Average		**0.62**	**7.93**	**0.70**	**9.44**

Table A.1: Initial concentrations Z_{tail} and Z_{total} classifications for selected proteins in Erk/Mek system. Classification is done using the threshold of ≥ 0.4.

Tables A.2 and A.3 shows the classification for the G2AR system for the topol-

ogy incompletion consensus profile. The classification is based on the removal of 6 nodes/edges respectively. Internalization protein *Rgi* classifies as (ii).

Name	Class	CPN		SPN	
	≥ 0.4	Z_{tail}	Z_{total}	Z_{tail}	Z_{total}
Rg*Ls	(iii)	0.0	1.29	-0.43	-0.7
Rs	(i)	-0.97	-0.93	-1.18	-1.35
ArrRg*Ls	(iii)	-0.96	-0.79	0.32	0.85
Rgi	(ii)	1.32	-0.39	1.51	0.95
R*Ls	(iii)	0.62	0.82	-0.22	0.24
Average		**1.10**	**8.88**	**1.02**	**14.84**

Table A.2: Topology (node removal) Z_{tail} and Z_{total} classifications for selected proteins in β2AR system. Classification is done using the threshold of ≥ 0.4.

Name	Class	CPN		SPN	
	≥ 0.4	Z_{tail}	Z_{total}	Z_{tail}	Z_{total}
Rg*Ls	(i)	-1.05	-0.74	-0.3	-0.35
Rs	(i)	-0.47	-0.84	-1.12	-1.4
ArrRg*Ls	(iii)	-0.01	0.4	0.25	0.75
Rgi	(ii)	1.64	1.56	1.56	1.15
R*Ls	(i)	-0.11	-0.37	-0.39	-0.15
Average		**1.59**	**7.36**	**1.09**	**5.50**

Table A.3: Topology (edge removal) Z_{tail} and Z_{total} classifications for selected proteins in β2AR system. Classification is done using the threshold of ≥ 0.4.

Table A.4 shows the classification for the Erk/Mek system for the 100% initial concentration inaccuracies consensus profile. $AKT(u)$, that classifies to (ii), shows the greatest amount of error in the consensus profile. Overall, all Z-scores are within one standard deviation and the classifications mostly indicate shifting in trajectories.

Name	Class	CPN		SPN	
	≥ 0.4	Z_{tail}	Z_{total}	Z_{tail}	Z_{total}
AKT(u)	(i)	0.07	0.39	0.77	0.63
AKT*	(i)	0.23	0.01	-0.44	-0.15
AKT	(i)	-0.34	-0.47	0.14	0.09
IRS1(u)	(i)	-0.29	-0.3	0.22	0.15
IRS1	(i)	-0.37	-0.49	-0.37	-0.42
IRS1*	(i)	-0.17	-0.27	-0.44	-0.46
AMPK	(i)	-0.37	-0.46	-0.34	-0.4
AMPK*	(i)	-0.22	-0.11	-0.3	-0.27
MAPK	(ii)	0.22	-0.47	-0.39	-0.31
MAPK*	(i)	0.16	0.06	0.27	0.17
PTEN	(i)	-0.35	-0.39	-0.38	-0.4
EGFR*	(i)	-0.37	-0.47	-0.45	-0.37
Average		**3.98**	**33.98**	**6.94**	**37.19**

Table A.4: Initial concentrations Z_{tail} and Z_{total} classifications for selected proteins in Erk/Mek system. Classification is done using the threshold of ≥ 0.4.

Tables A.5 and A.6 shows the classification for the Erk/Mek system for the topology incompletion consensus profile. The classifications are based on the removal of 6 nodes/edges.

Name	Class	CPN		SPN	
	≥ 0.4	Z_{tail}	Z_{total}	Z_{tail}	Z_{total}
AMPK*	(i)	-0.78	-0.84	-0.62	-0.89
AMPK	(i)	-0.78	-0.84	-0.62	-0.89
AKT(u)	(ii)	1.27	0.8	1.65	1.47
AKT*	(iii)	0.89	1.33	0.25	0.49
EGFR*	(iii)	-0.6	-0.43	-0.66	-0.19
Average		**1.81**	**14.35**	**1.48**	**22.27**

Table A.5: Topology (node removal) Z_{tail} and Z_{total} classifications for selected proteins in Erk/Mek system. Classification is done using the threshold of ≥ 0.4.

Name	Class	CPN		SPN	
	≥ 0.4	Z_{tail}	Z_{total}	Z_{tail}	Z_{total}
AMPK*	(i)	-0.88	-0.83	-0.67	-0.84
AMPK	(i)	-0.88	-0.83	-0.67	-0.84
AKT(u)	(i)	1.17	1.22	1.65	1.48
AKT*	(i)	0.97	0.94	0.25	0.54
EGFR*	(i)	-0.38	-0.49	-0.55	-0.33
Average		**1.53**	**17.13**	**1.58**	**25.35**

Table A.6: Topology (edge removal) Z_{tail} and Z_{total} classifications for selected proteins in Erk/Mek system. Classification is done using the threshold of ≥ 0.4.

Bibliography

[1] J. Fisher and T. A. Henzinger, "Executable cell biology," *Computational Biology*, vol. 25, no. 11, pp. 1239–1249, November 2007.

[2] D. Gillespie, "A general method for numerically simulating stochastic time evolution of coupled chemical reactions," *Journal of Computational Physics*, vol. 22, pp. 403–434, 1976.

[3] V. N. Reddy, M. L. Mavrovouniotis, and M. N. Liebman, "Petri net representation in metabolic pathways," in *ISMB-93*. AAAI, 1993.

[4] M. Heiner and I. Koch, "Petri net based models validation in systems biology," in *ICATPN*, ser. LNCS. Springer-Verlag Berlin Heidelberg, 2004, pp. 216–237.

[5] A. Varma and B. O. Palsson, "Stoichiometric flux balance models quantitatively predict growth and metobolic by-product secretion in wild-type Escherichia coli w3110," *Applied and Environmental Microbiology*, vol. 60, no. 10, pp. 3724–3731, October 1994.

[6] V. N. Reddy, M. N. Liebman, and M. L. Mavrovouniotis, "Qualitative analysis of biochemical reaction systems," *Computational Biology*, vol. 26, no. 1, pp. 9–24, 1996.

[7] K. Voss, M. Heiner, and I. Koch, "Steady state analysis of matabolic pathways using Petri nets," *In Silico Biology*, vol. 3, pp. 367–387, September 2003.

[8] H. Genrich, R. Kuffner, and K. Voss, "Executable Petri net models for the analysis of metabolic pathways," in *International Journal of software tools for technology transfers (STTT)*. SpringerLink, 2001, vol. 3, no. 4, pp. 394–404.

[9] R. Hofestadt and S. Thelen, "Quantative modeling of biochemical networks," *In Silico Biology*, vol. 1, pp. 39–53, 1998.

[10] W. Samarrai, I. Barjis, J. W. Yeol, and Y. Ryu, "Modeling of carbohydrate metabolism: From dietary carbohydrate to pyruvic acid by Petri-nets (PN)," in *Bioengineering Conference, 2005. Proceedings of the IEEE 31st Annual Northeast*, 2005, pp. 267–268.

[11] L. Popova-Zeugmann, M. Heiner, and I. Koch, "Modelling and analysis of biochemical networks with timed Petri nets," in *13th International Workshop on Concurrency Specification and Programming*, vol. 170, 2004, p. 143.

[12] S. A. Kauffman, "Metabolic stability and epidenesis in randomly constructed genetic nets," *J. Theoret. Biol*, vol. 22, pp. 437–467, 1969.

[13] R. Thomas, "Kinetic logic: a Boolean approach to the analysis of complex regulatory systems," *Notes Biomath*, no. 29, 1979.

[14] T. Akutsu and S. Miyano, "Identification of genetic networks from a small number of gene expression patterns under the boolean network model," *Pacific Symposium on Biocomputing*, vol. 4, pp. 17–28, 1999.

[15] A. Silvescu and V. Honavar, "Termporal Boolean network models of generic networks and their inference from gene expression time series," *Complex Systems*, vol. 11, 1997.

[16] J. Steggles, R. Banks, O. Shaw, and A. Wipat, "Modeling and analyzing genetic networks: From Boolean networks to Petri nets," in *CMSB'06, LNCS 4210*. Springer, 2006, pp. 127–141.

[17] C. Chaouiya, E. Remy, P. Ruet, and D. Thieffry, "Qualitative modelling of genetic networks: From logical regulatory graphs to standard petri nets," in *LNCS*. Springer-Verlag, 2004, pp. 137–156.

[18] C. Chaouiya, E. Remy, and D. Thieffry, "Petri net modelling of biological regulatory networks," in *Proceedings of CMSB-3*. Thieffry, 2005.

[19] R. Banks, L. J. Steggles, R. Banks, and L. J. Steggles, "A high-level petri net framework for multi-valued genetic regulatory networks," *Journal of Integrative Bioinformatics*, 2007.

[20] H. D. Jong, "Modeling and simulation of genetic regulator systems: a litrature review," *Journal of Computational Biology*, vol. 9, no. 1, pp. 67–103, 2002.

[21] R. I. Hamed, S. I. Ahson, and R. Parveen, "Designing genetric regulation networks using fuzzy petri nets approach," *International Journal of Automation and Computin*, vol. 7, no. 3, pp. 403–412, 2010.

[22] C. Chaouiya, E. Remy, and D. Thieffry, "Petri net modelling of biological regulatory networks," *Journal of Discrete Algorithms*, pp. 165–177, June 2008.

[23] C. Li, Q.-W. Ge, M. Nakata, H. Matsuno, and S. Miyano, "Modeling and simulation of signaling transduction in an apoptosis pathway by using Petri nets," *J Biosci*, vol. 32, no. 1, pp. 113–127, January 2007.

[24] D. Ruths, L. Nakhleh, and P. Ram, "Rapidly exploring structural and dynamic properties of signaling networks using pathwayoracle," *BMC Systems Biology*, vol. 2, no. 76, August 2008.

[25] I. Koch, W. Reisig, and F. Schreiber, Eds., *Modeling in Systems Biology: The Petri net approach.* Springer, 2011.

[26] T. Ideker, T. Galitski, and L. Hood, "A new approach to decoding life: Systems biology," *Annual review of genomics and human genetics*, vol. 2, no. 1, pp. 343–372, 2001.

[27] J. Ackermann and I. Koch, *Modeling in Systems Biology: The Petri net approach.* Springer-Verlag London Limited, 2011, computational biology Quantitative Analysis, pp. 153–178.

[28] C. A. Petri, "Kommunikation mit automaten (communication with automata)," Ph.D. dissertation, University of Bonn, 1962.

[29] J. Peterson, *Petri net theory and the modeling of systems.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 1981.

[30] A. Bobbio, "System modeling with Petri nets," Ph.D. dissertation, Instituto Elettrotecnico Nazionale Galileo Ferraris, Torino, Italy, 1990.

[31] C. Chaouiya, "Petri net modeling of biological networks," *Briefings in Bioinformatics*, vol. 8, no. 40, pp. 210–219, July 2007.

[32] D. Gillespie, "Exact stochastic simulation of coupled checmical reactions," *The Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, May 1977.

[33] D. Gilbert and M. Heiner, "From Petri nets to differential equations - an integrative approach for biochemical network analysis," University of Glasgow, Glasgow G12 8QQ UK, Tech. Rep. 2008, December 2005.

[34] Y. Cao, H. Li, and L. Petzold, "Efficient formulation of the stochastic simulation algorithm for chemically reacting systems," *J Chem Phys*, vol. 121, no. 4059, 2004.

[35] D. A. McQuarrie, "Stochastic approach to chemical kinetics," *J Appl. Porb.*, vol. 4, pp. 413–478, 1967.

[36] D. Gillespie, "A rigorous derivation of the chemical master equation," *Physica*, vol. 188, pp. 404–425, 1992.

[37] H. H. McAdams and A. Arkin, "Stochastic mechanism in gene expression," *Proc Natl Acad Sci*, vol. 94, pp. 814–819, February 1997.

[38] A. Arkin, J. Ross, and H. H. McAdams, "Stochastic kinetic analysis of development pathway bifurcation in Phage lamda-Infected *Escherichia coli* cell," *Genetics*, vol. 149, pp. 1633–1648, August 1998.

[39] M. A. Gibson and J. Bruck, "Efficient exact stochastic simulation of chemical systems with many species and many channels," *J Phys Chem*, vol. 104, no. 9, February 2000.

[40] D. T. Gillespie, "Approximate accelerated stochastic simulation of chemically reaction systems," *J Chem Phys*, vol. 115, no. 4, pp. 1716–1733, July 2001.

[41] ——, *Lecture Notes in Computer Science.* Springer-Verlag Berlin Heidelberg, 2008.

[42] T. Tian and K. Burrage, "Binomial leap methods for simulating stochastic chemical kinetics," *J Chem Phys*, vol. 121, no. 21, pp. 10 356–10 364, December 2004.

[43] A. Chatterjee and D. G. Vlachos, "Binomial distribution based tau-leap accelerated stochastic simulation," *J Chem Phys*, vol. 122, January 2005.

[44] Y. Cao, D. T. Gillespie, and L. Petzold, "Avoiding negative populations in explicit poisson tau-leaping," *J Chem Phys*, vol. 123, August 2005.

[45] ——, "Efficient step size selection for the tau-leaping simulation method," *J Chem Phys*, vol. 124, January 2006.

[46] S. J. Vayttaden, J. Friedman, T. M. Tran, T. C. Rich, C. W. Dessauer, and R. B. Clark, "Quantitative modeling of grk-mediated beta2ar regulation," *PLoS Computational Biology*, vol. 6, no. 1, January 2010.

[47] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer, "Copasi — a complex pathway simulator," *Bioinformatics*, vol. 22, pp. 3067–74, 2006.

[48] L. Petzold, "Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations," *SIAM J. Sci. Stat. Comput.*, vol. 4, pp. 136–148, 1983.

[49] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *JOURNAL OF KNOWLEDGE AND INFORMATION SYSTEMS*, vol. 3, pp. 263–286, 2000.

[50] S. Iadevaia, Y. Lu, F. C. Morales, G. B. Mills, and P. Ram, "Identification of optimal drug combinations targeting cellular networks: Integrating phosphoproteomics and computational network analysis," *Cancer Research*, vol. 70, no. 17, September 2010.

[51] U. S. Bhalla and R. Iyengar, "Emergent properties of network of biological signaling pathways," *Science*, vol. 283, pp. 381–387, January 1999.

[52] L. H. Leemis and S. K. Park, *Discrete-Event simulations: A first couse.* Prentice Hall, 2006, ch. 8.

[53] (2010). [Online]. Available: http://code.google.com/p/jmotif/wiki/SAX

[54] D. Q. Goldin and P. C. Kanellakis, "On similarity queries for time-series data: Constraint specification and implementation," in *Proceedings of the Fist International Conference on Principles and Practice of Constraint Programming.* Springer-Verlag London, 1995.

[55] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *DMKD Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, ACM, Ed. New York, NY: ACM, 2003.

[56] M. Weber and E. Kindler, *The Petri Net Kernel.* Springer-Verlag Berlin Heidelberg, 2003, pp. 109–124.

[57] R. Albert and H. G. Othmer, "...but no kinetic details needed," *SIAM News*, vol. 36, no. 10, 2003.

[58] D. Ruths, M. Muller, J.-T. Tseng, L. Nakhleh, and P. Ram, "The signaling Petri net-based simulator: A non- parametric strategy for characterizing the dynamics of cell-specific signaling networks," *PLoS Computational Biology*, vol. 4, no. 2, February 2008.

[59] J. M. Lee, E. G. Gianchandani, J. A. Eddy, and J. A. Papin, "Dynamic analysis of integrated signaling, metabolic, and regulatory networks," *PLoS Computational Biology*, vol. 4, no. 5, May 2008.

[60] T. Shlomi, Y. EisenBerg, R. Sharan, and E. Ruppin, "A genome-scale computational study of the interplay between transcriptional regulation and metabolism," *Molecular Systems Biology*, vol. 3, no. 101, February 2007.

[61] M. W. Covert, N. Xiao, T. J. Chen, and J. R. Karr, "Integrating metabolic, transciptional regulatory and signal transduction models in Escherichia coli," *Bioinformatics*, vol. 24, no. 18, pp. 2044–2050, July 2008.